UNIVERSITÄT BONN

Holistic Video Understanding: Spatio-Temporal Modeling and Efficiency

DISSERTATION

zur Erlangung des Doktorgrades (*Dr. rer. nat.*)

der Mathematisch-Naturwissenschaftlichen Fakultät

der Rheinischen Friedrich–Wilhelms–Universität, Bonn

vorgelegt von

MOHSEN FAYYAZ
aus
Tehran, Iran

Bonn, 2024

Angefertigt mit Genehmigung der Mathematisch-Naturwissenschaftlichen Fakultät der Rheinischen Friedrich–Wilhelms–Universität Bonn

1. Gutachter / 1st Advisor: Prof. Dr. Juergen Gall
2. Gutachter / 2nd Advisor: Prof. Dr. Cees Snoek
Tag der Promotion / Day of Promotion: 18.03.2024
Erscheinungsjahr / Year of Publication: 2024

Abstract

by Mohsen Fayyaz

for the degree of

Doctor rerum naturalium

Video understanding, a subset of computer vision, has gained significant attention due to the increasing consumption of video data. In particular, action recognition, the task of classifying human actions depicted in trimmed video clips, has seen substantial progress with the advent of deep neural networks. 3D convolutional neural networks (CNNs), which exploit temporal cues in addition to spatial cues, have demonstrated promising results. However, these networks have limitations in capturing hidden information in both spatial and temporal correlations between channels, leading to decreased performance. To address this issue, we introduce a novel network architecture block that efficiently captures both spatial-channel and temporal-channel correlation information throughout the network layers.

Additionally, training 3D CNNs requires extensive labeled datasets, increasing computational cost and time, and limiting the performance of these architectures. To mitigate this problem, we propose an effective supervision transfer method that allows a 2D CNN pre-trained on ImageNet to act as a "teacher" for the stable weight initialization of a randomly initialized 3D CNN, avoiding the need for extensive training from scratch.

The lack of established video benchmarks that integrate the joint recognition of multiple semantic aspects in dynamic scenes remains a challenge in the field of video understanding. To address this, we propose the "Holistic Video Understanding" (HVU) dataset, a hierarchical, multi-label, multi-task, large-scale video benchmark with comprehensive tasks and annotations for video analysis and understanding. Furthermore, motivated by the goal of holistic representation learning, we introduce a new spatio-temporal architecture for the task of video classification. Our architecture focuses on multi-label, multi-task learning, jointly solving multiple spatio-temporal problems simultaneously by fusing 2D and 3D architectures in order to capture both spatial and temporal information.

The field of video processing has seen significant growth with the advent of 3D convolutional neural networks (3D CNNs), which are highly effective for action classification tasks. However, these networks are often resource-intensive to train and deploy for inference. In this work, we present an approach that aims to improve the efficiency of 3D CNNs for both training and inference by dynamically adapting the temporal resolution of the network to the input frames. By exploiting the redundancy within temporal features, our method allows the 3D CNN to select the most valuable and informative temporal features for the action classification task, while avoiding the waste of computational resources on redundant information. In addition to 3D CNNs, vision transformers have recently demonstrated effectiveness in image and video classification tasks. However, the high computational cost of their transformer blocks can be prohibitive for deployment on edge devices. To address this issue, we present a method for efficiently reducing the number of tokens in a vision transformer by automatically selecting an appropriate number of tokens at each stage based on the content of the image or video. Through this approach, we aim to enable transformers to dynamically adapt their computational resources to efficiently process each input image or video.

Despite the numerous methods that have achieved state-of-the-art results in video understanding on trimmed video clips, real-world scenarios often involve untrimmed videos with multiple actions requiring recognition at each frame. This task, referred to as temporal action segmentation, involves not only recognizing actions within untrimmed videos but also determining their order and duration. While there has been a surge of interest in temporal action segmentation, annotating each frame of a video is a tedious and costly process. As a result, weakly supervised approaches have been developed to learn temporal action segmentation from videos that are only weakly labeled. In such cases, the set of actions present in a video is obtained as weak supervision, providing a cost-effective means of annotating a video. Given a set of actions in which only the list of actions occurring in the video is known, but not when, how often, or in which order they occur, we propose an end-to-end trainable approach to train a temporal action segmentation model with such weak supervision. Our approach divides the video into smaller temporal regions and predicts for each region the action label and its length, as well as estimating action labels for each frame. By measuring the consistency of the frame-wise predictions with respect to the temporal regions and annotated action labels, the network learns to divide the video into class-consistent regions. We evaluate our approach on three datasets and achieve state-of-the-art results. In spite of the cost-effectiveness of utilizing a set of actions for training a model, using it as proper supervision for temporal action segmentation models presents a daunting challenge. Given the limitations of approaches with weak actions set supervision, methods that can leverage stronger forms of supervision are of great importance. One such type of weak supervision is transcripts, which are ordered lists of actions indicating the order in which actions occur in a training video, but not when they occur. As a culminating contribution, we propose a novel end-to-end framework for weakly supervised action segmentation using a two-branch neural network. The network's dual branches independently predict redundant yet distinct action segmentation representations, and we incorporate a mutual consistency loss term to enforce consistency between these redundant representations. Our approach attains the accuracy of state-of-the-art methods while exhibiting marked improvements in efficiency.

Keywords: video understanding, action recognition, efficient spatio-temporal convolutional neural networks, efficient vision transformers, temporal action segmentation

Acknowledgements

I would like to extend my sincerest gratitude to my advisor, Professor Juergen Gall, for providing unwavering guidance and support throughout the entirety of my Ph.D. journey. His willingness to allow me the freedom to pursue my own research interests and his constant encouragement have been invaluable to my progress.

I am also deeply appreciative of my colleagues and friends within the Computer Vision Group at the University of Bonn for creating a warm and welcoming environment. Special thanks go to Yaser for his kind welcome upon my initial arrival in Bonn and to Sovan for his assistance in securing accommodation and providing support during my early days in the city.

I am indebted to my co-authors, Ali, Vivek, Yaser, Emad, Soroush, Farnoush, Nadine, and Saber, for their invaluable contributions and efforts that made the work we accomplished possible.

I would also like to express my gratitude to Mehdi Noroozi from Bosch Center for Artificial Intelligence for the enlightening internship and PhD sabbatical experience, and his support during the COVID-19 lockdown to navigate bureaucratic obstacles. It was a privilege to collaborate with such a talented research scientist.

I would like to thank Eric Sommerlade from Microsoft for his support during my internship. His belief in my research idea and his valuable guidance have been instrumental in my progress. I am grateful for the valuable lessons I have learned during this experience.

Lastly, I would like to extend my heartfelt thanks to my family. My wife and partner, Farnoush, has been a constant source of unwavering support, patience, care, and kindness throughout the years. Her sacrifices have made this achievement possible. I am also thankful to my parents and brother for their selfless support throughout my life and especially during my Ph.D. journey. Thank you.

Contents

List of Figures

List of Tables

Nomenclature

Abbreviations

An alphabetically sorted list of abbreviations used in the thesis:

1D	1 Dimensional
2D	2 Dimensional
3D	3 Dimensional
CNN	Convolutional Neural Network
GFLOP	Giga Floating Point Operations
GPU	Graphical Processing Unit
GRU	Gated Recurrent Unit
IDT	Improved Dense Trajectories
mAP	Mean Average Precision
MLP	Multi Layer Perceptron
MoF	Mean over Frames
PCA	Principal Component Analysis
ReLU	Rectified Linear Unit
RGB	Red Green Blue (3 Channels for a color image)
RNN	Recurrent Neural Network
SGD	Stochastic Gradient Descent

List of Publications

The thesis is based on the following publications:

- **Spatio-Temporal Channel Correlation Networks for Action Classification**
  Ali Diba*, Mohsen Fayyaz*, Vivek Sharma, Mohammad Arzani, Rahmah Yousefzadeh, Juergen Gall, and Luc Van Gool
  European Conference on Computer Vision (ECCV), 2018.
  doi: 10.1007/978-3-030-01225-0_18
  (* denotes equal contribution)

- **Large Scale Holistic Video Understanding**
  Ali Diba*, Mohsen Fayyaz*, Vivek Sharma*, Manohar Paluri, Juergen Gall, Rainer Stiefelhagen, and Luc Van Gool
  European Conference on Computer Vision (ECCV), 2020.
  doi: 10.1007/978-3-030-58558-7_35
  (* denotes equal contribution, listed in alphabetical order)

- **SCT: Set Constrained Temporal Transformer for Set Supervised Action Segmentation**
  Mohsen Fayyaz, and Juergen Gall
  IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2020.
  doi: 10.1109/CVPR42600.2020.00058

- **3D CNNs with Adaptive Temporal Feature Resolutions**
  Mohsen Fayyaz*, Emad Bahrami*, Ali Diba, Mehdi Noroozi, Ehsan Adeli, Luc Van Gool, and Juergen Gall
  IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2021.
  doi: 10.1109/CVPR46437.2021.00470
  (* denotes equal contribution)

- **Long Short View Feature Decomposition via Contrastive Video RepresentationLearning**
  Nadine Behrman, Mohsen Fayyaz, Juergen Gall, and Mehdi Noroozi
  IEEE/CVF International Conference on Computer Vision and Pattern Recognition (ICCV), 2021.
  doi: 10.1109/ICCV48922.2021.00911

- **Fast Weakly Supervised Action Segmentation Using Mutual Consistency**
  Yaser Souri*, Mohsen Fayyaz*, Luca Minciullo, Gianpiero Francesca, and Juergen Gall
  IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI), 2022.

doi: 10.1109/TPAMI.2021.3089127
(* denotes equal contribution)

- **Adaptive Token Sampling For Efficient Vision Transformers**
  Mohsen Fayyaz*, Soroush Abbasi Koohpayegani*, Farnoush Rezaei Jafari*, Sunando Sengupta, Hamid Reza Vaezi Joze, Eric Sommerlade, Hamed Pirsiavash, and Juergen Gall
  European Conference on Computer Vision (ECCV), 2022.
  doi: 10.1007/978-3-031-20083-0_24
  (* denotes equal contribution)

- **TaylorSwiftNet: Taylor Driven Temporal Modeling for Swift Future Frame Prediction**
  Saber Pourheydari*, Emad Bahrami*, Mohsen Fayyaz*, Gianpiero Francesca, Mehdi Noroozi, and Juergen Gall
  British Machine Vision Conference (BMVC), 2022.
  doi: 10.48550/arXiv.2110.14392
  (* denotes equal contribution)

CHAPTER 1

Introduction

Contents

1.1 Motivation

As humans continue to strive for automation in various fields, the development of intelligent agents capable of performing complex tasks has become increasingly important. In order to teach machines to function in a manner similar to humans, it is necessary to find efficient methods of training them. *LeCun* (2022) has emphasized the importance of good perception, particularly through vision, in the development of intelligent agents. Given the vital role that vision plays in our own understanding of the world, it is no surprise that computer vision has become a key focus in the field of machine learning. The increasing availability and consumption of video data has led to a significant interest in the field of video understanding within the research community. The ability to analyze, monitor, annotate, and learn from videos is crucial for the development of machine learning systems that can effectively perceive and understand the world.

The field of video understanding has long been preoccupied with classifying human actions depicted in trimmed video clips, also referred to as action recognition (*Poppe*, 2010). In the era of deep neural networks, significant advancements have been made in the performance of action recognition techniques. Initially, these techniques struggled to match the performance of hand-crafted features for video classification (*Karpathy et al.*, 2014a). However, this changed with the introduction of the two-stream network (*Simonyan and Zisserman*, 2014), which marked a turning point in the field, as deep learning models began to achieve performance on par with state-of-the-art hand-crafted features. This architecture consists of two parallel convolutional neural networks (CNNs) processing RGB frames and stacks of optical flow frames, respectively, which then fuse their outputs and pass them to a classifier. This marked a turning point in the field, as deep learning models began to achieve performance on par with state-of-the-art hand-crafted features. Subsequent methods have built upon this two-stream architecture, resulting in further improvements. To address the challenges of designing and training 3D CNNs from scratch, *Carreira and Zisserman* (2017) proposed the I3D network, which inflates a successful 2D CNN architecture into 3D and initializes the 3D CNN weights using pre-trained weights from the ImageNet dataset (*Deng et al.*, 2009). The exploitation of temporal cues rather than merely spatial cues has been shown to have compelling advantages for video classification (*Diba et al.*, 2017; *Tran et al.*, 2015; *Yue-Hei Ng et al.*, 2015), and recent works have focused on improving the modeling of spatio-temporal correlations. However, 3D CNNs have limitations in capturing hidden information in both spatial and temporal correlations between channels, leading to decreased performance. To address this issue, we propose a new network architecture block that efficiently captures both spatial-channel and temporal-channel correlation information throughout network layers. In addition to these technical challenges, another major issue in the use of 3D CNNs is the need for extra large labeled datasets for training. This not only increases the computational cost and time required but also limits the performance of these architectures. To address this issue, we propose an effective supervision transfer method that allows a 2D CNN pre-trained on ImageNet to act as a teacher for the stable weight initialization of a randomly initialized 3D CNN, avoiding the need for extensive training from scratch.

Despite significant progress in spatio-temporal models for video understanding, most of these models primarily focus on action recognition. Consequently, video understanding as a more generic problem, which encompasses the recognition of multiple semantic aspects such as scenes or environments, objects, actions, events, attributes, and concepts remains a challenging issue in computer vision. This challenge stems from the absence of established video benchmarks that integrate the joint recognition of multiple semantic aspects in dynamic scenes. Although deep neural networks have facilitated substantial advancements in various sub-fields of computer vision, a common drawback when training these networks for video understanding with a single label per task is the inability to adequately describe the content of a video. This limitation hampers neural networks' potential to learn a generic feature representation for comprehensive and complex video analysis. To address this issue, this thesis proposes recasting the video understanding problem as multi-task classification, where multiple labels are assigned to a video from multiple semantic aspects. This approach allows for the learning of a generic feature representation for video analysis and understanding, similar to the way that image classification deep neural networks trained on ImageNet facilitate the learning of a generic feature representation for various vision tasks. To this end, we introduce the "Holistic Video Understanding" dataset (HVU), a hierarchical, multi-label, multi-task, large-scale video benchmark with comprehensive tasks and annotations for video analysis and understanding. Motivated by the goal of holistic representation learning, we also propose a new spatio-temporal architecture for the task of video classification. Our architecture focuses on multi-label, multi-task learning to jointly solve multiple spatio-temporal problems simultaneously, fusing 2D and 3D architectures into one by combining intermediate representations of appearance and temporal cues to achieve a robust spatio-temporal representation.

Although extending the capabilities of 3D CNNs beyond action recognition enhances video representation and understanding, these networks are often resource-intensive for both training and deployment. To address this issue, some researchers have proposed using multiple CNNs with different levels of complexity to process different parts of the video (*Korbar et al.*, 2019; *Zhu et al.*, 2020). While these approaches can reduce the number of floating point operations (FLOPs) during inference, they increase training time and do not address the inherent inefficiency of 3D CNNs in processing certain types of input. In this thesis, we present an approach that aims to make 3D CNNs more efficient for both training and inference. Our approach is based on the observation that the computational cost of a 3D CNN depends on the temporal resolution at which it operates at each stage of the network. While the temporal resolution can vary at different stages, the schemes that define how it is reduced are hard-coded and apply the same to all videos. However, it is unlikely that a single scheme will be optimal for all videos, as a low temporal resolution may discard important information for some videos, while a high temporal resolution may result in redundant feature maps and increased computational time. To address this issue, we propose a method for dynamically adapting the temporal resolution of a 3D CNN to the input video content. By exploiting the redundancy within temporal features, our method allows the 3D CNN to process and select the most valuable and informative temporal features for the action classification task while avoiding the waste of computational resources on redundant information. In contrast to previous approaches, our method dynamically adapts the temporal resolution within the network, ensuring that important information is not discarded and computational resources are used efficiently.

Following the success of the CNNs in image and video understanding, recently vision transformers have demonstrated their effectiveness in image classification tasks compared to traditional CNNs (*Dosovitskiy et al.*, 2021; *Touvron et al.*, 2021; *Jiang et al.*, 2021; *Wu et al.*, 2021). These models have also shown promise in action recognition tasks (*Bertasius et al.*, 2021; *Liu et al.*, 2021; *Bulat et al.*, 2021). While vision transformers have a higher representation power, the computational cost of their transformer blocks can be prohibitively high for deployment on edge devices. To mitigate the high computational cost, DynamicViT (*Rao et al.*, 2021a) employs a token scoring neural network to predict which tokens are redundant, keeping a fixed ratio of tokens at each stage. However, this approach introduces additional computational overhead, requiring the training of an additional network and modification of the loss function with additional hyperparameters. To alleviate these limitations, EViT (*Liang et al.*, 2022) utilizes attention weights as tokens' importance scores. Both DynamicViT and EViT require re-training when the fixed target ratios need to be changed, which restricts their flexibility in deployment. In this thesis, following our study on the efficiency of 3D CNNs, we present a method for efficiently reducing the number of tokens in a vision transformer without the aforementioned limitations. Our approach is motivated by the observation that in image and action classification tasks, not all parts of the input image or video contribute equally to the final classification scores and some parts may contain redundant or irrelevant information. The amount of relevant information varies based on the content of the image or video. To address this issue, we propose a method that automatically selects an appropriate number of tokens at each stage based on the content of the image or video, allowing the number of selected tokens at each stage to vary for different images or videos. Through this approach, we aim to address the question of how a transformer can dynamically adapt its computational resources to efficiently process each input image or video.

Despite the numerous methods that have achieved state-of-the-art results in video understanding on trimmed video clips, real-world scenarios often involve untrimmed videos containing multiple actions, requiring the recognition of actions for every given frame. This task is known as temporal action segmentation, which involves the recognition of actions within untrimmed videos in addition to their order and duration. Temporal action segmentation is critical for effectively analyzing untrimmed videos frame-by-frame to identify and classify actions within them. It has numerous practical applications, including in robotics, search engines, and factory assembly line quality control. In recent years, several state-of-the-art models have been proposed for this task (*Kuehne et al.*, 2016b; *Lea et al.*, 2017; *Abu Farha and Gall*, 2019), which have achieved impressive results by leveraging techniques such as multi-scale sliding window processing, Markov models, and temporal convolutions to capture long-range temporal dependencies. However, these models are typically trained in a fully supervised setting, requiring each training video to be fully annotated with frame-wise labels indicating the action classes present at each frame. This annotation approach, while effective, is often impractical due to the cost and time required for such extensive annotation. As a result, there is a need for methods that can handle the challenges of limited or noisy annotations and can be trained in a more efficient and scalable manner. In this thesis, we aim to address this issue by exploring methods for temporal action segmentation that can handle various forms of weak supervision, including ordered sequence of actions without the actions' duration, and even set of actions without orders or duration of the actions.

Temporal action segmentation in untrimmed videos is highly sought after in real-world scenarios. While fully annotating video frames can be a straightforward way to train temporal action segmentation models, it is also time-consuming and costly. As a more cost-effective alternative, obtaining a set of actions present in a video can be used for annotation; however, this approach makes training a traditional temporal action segmentation model challenging due to the lack of supervision. *Richard et al.* (2018a) proposed learning temporal action segmentation solely from a set of action labels provided for a complete video spanning several minutes. In this scenario, the actions that occur are known, but not their timing, order, or frequency, making the task notably more difficult compared to learning from ordered action sequences, also known as transcripts, or fully supervised learning. To tackle this challenge, *Richard et al.* (2018a) suggests generating hypothesized transcripts that include each action label of a video at least once and then inferring frame-level labeling through alignment with these hypothesized transcripts. Although this approach demonstrates the feasibility of learning from weak annotations, even for lengthy videos, it does not directly solve the problem. Instead, it transforms it into a weakly supervised learning problem with multiple hypothesized transcripts per video. This method is inefficient, as aligning all possible transcripts generated from a set of action labels is infeasible, and it does not utilize the provided annotations directly for learning. To address these limitations, we propose a method that directly incorporates the action labels given for each training video into the loss function, enabling end-to-end training of the model.

While obtaining a set of actions occurring in a video is among the most economical means of annotating videos, using it as proper supervision for temporal action segmentation models is highly challenging. As a result, approaches utilizing only a set of actions as a source of supervision are still in their infancy in terms of performance. Given the immaturity of approaches with weak supervision, methods that can utilize stronger forms of supervision are of significant importance. In particular, ordered lists of actions known as transcripts, which describe the order in which actions occur in a training video but not when they occur, are a popular type of weak supervision (*Huang et al.*, 2016; *Richard et al.*, 2017, 2018b; *Ding and Xu*, 2018; *Chang et al.*, 2019; *Li et al.*, 2019). To learn from transcripts, previous approaches have attempted to align transcripts with training videos, inferring frame-wise labels for each video based on the provided transcripts, which are then used as pseudo ground truth for training. The Viterbi algorithm is commonly employed for this alignment process, which takes the estimated frame-wise class probabilities for a video and determines the optimal sequence of frame labels that does not violate the action order specified in the transcript.

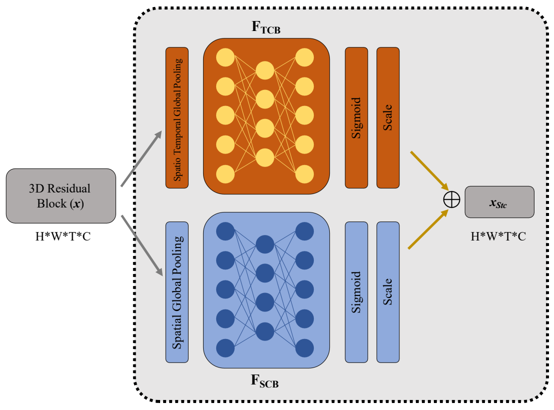
While *Richard et al.* (2017) and *Ding and Xu* (2018) perform alignment after each epoch for all training videos, *Richard et al.* (2018b) and *Li et al.* (2019) apply it at each iteration to a single video. During inference, previous approaches, with the exception of ISBA (*Ding and Xu*, 2018), rely on segmentation through alignment, searching over all transcripts in the training set to identify the one that best aligns with the test video, resulting in inefficient performance as the number of transcripts in the training set increases. In contrast, ISBA is fast but does not achieve the accuracy of state-of-the-art approaches. This means that currently, one must choose between accurate or fast approaches, but cannot have both. In this thesis, we make contributions to the field of temporal action segmentation with a focus on weak supervision and training and testing efficiency, proposing a novel approach for efficient, end-to-end weakly supervised temporal action segmentation. We address the gap in previous works by proposing an approach that is nearly as fast as ISBA and nearly as accurate as state-of-the-art methods. Instead of optimizing over all possible transcripts during inference, our approach directly predicts the transcript for a video as well as the frame-wise class probabilities, allowing for the direct prediction of the optimal label sequence from the estimated transcript and frame-wise class probabilities without dependence on the number of transcripts used for training.

1.2 Contributions

In this thesis, we present a suite of approaches that address the complex and multifaceted challenges of video understanding discussed above. Our contributions, outlined below, focus on addressing the difficulties inherent in modeling spatio-temporal dependencies within video data, creating a large scale holistic video understanding dataset, making video understanding models more efficient, and training models with less supervision. By tackling these challenges, we aim to advance the field of video understanding and enable the more efficient and effective utilization of large video datasets for the development and evaluation of advanced deep learning models.

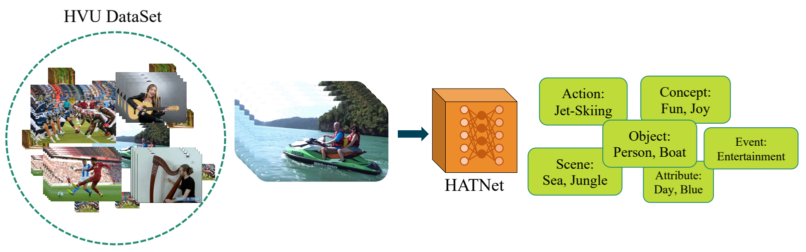
1.2.1 Spatio-Temporal Channel Correlation for Action Classification

Our first contribution is a new block, called ‘Spatio-Temporal Channel Correlation’ (STC), which models correlations between channels of a 3D CNN with respect to temporal and spatial features. By integrating this block into existing state-of-the-art architectures such as ResNet and ResNext (*Hara et al.*, 2018), we observe performance improvements of 2-3%. In addition to this contribution, we also present a technique for transferring knowledge from pre-trained 2D CNNs to randomly initialized 3D CNNs, allowing for the efficient fine-tuning of 3D CNNs with significantly reduced amounts of training data. Our approach outperforms both generic and preceding methods in the field, demonstrating the effectiveness of our proposed techniques for improving the performance of 3D CNNs in video understanding tasks.



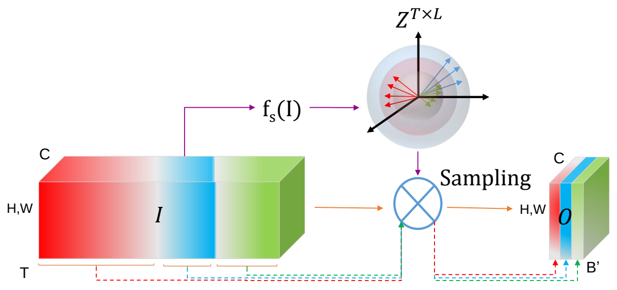
STC Block

1.2.2 Holistic Video Understanding



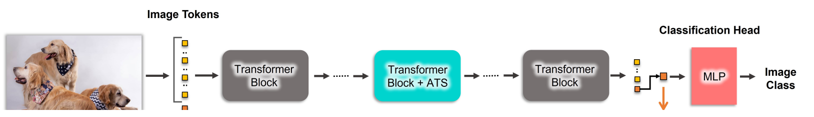
In previous research, we concentrated on techniques for enhancing the performance of 3D CNNs in the realm of action recognition. However, video understanding - a field encompassing the recognition of multiple semantic aspects, such as scene or environment, objects, actions, events, attributes, and concepts - remains a challenging problem. This is despite the notable progress made in action recognition. One contributing factor is the absence of well-established video benchmarks that integrate the simultaneous recognition of various semantic aspects within dynamic scenes. To address this gap, we introduce the "Holistic Video Understanding" Dataset (HVU), a large-scale dataset organized in a semantic taxonomy that focuses on multi-label and multi-task video understanding as a comprehensive problem. HVU contains approximately 572k videos with 9 million annotations spanning 3142 labels, encompassing categories of scenes, objects, actions, events, attributes, and concepts that capture real-world scenarios. We demonstrate the generalization capabilities of HVU on three challenging tasks: video classification, video captioning, and video clustering. For video classification, we propose the "Holistic Appearance and Temporal Network" (HATNet), a novel spatio-temporal deep neural network architecture that fuses 2D and 3D architectures by combining intermediate representations of appearance and temporal cues, and is trained in an end-to-end manner for multi-label and multi-task learning.

1.2.3 Adaptive Spatio-Temporal Convolutional Neural Networks for Efficient Video Classification



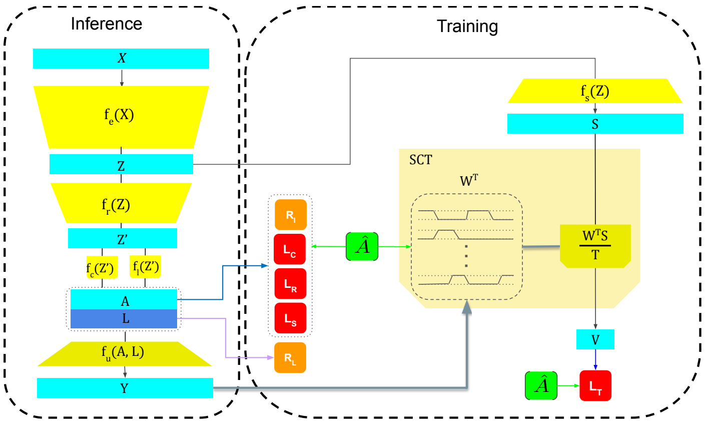
Despite efforts made in prior research to improve the performance and generalizability of video understanding models, deep video models are often computationally expensive to train and also for inference. While the computation cost of a 3D CNN can be reduced by decreasing the temporal feature resolution within the network, there is no setting that is optimal for all input clips. In light of this, we introduce a differentiable Similarity Guided Sampling (SGS) module that can be integrated into any existing 3D CNN architecture. The SGS module empowers 3D CNNs by learning the similarity between temporal features and grouping similar features together, resulting in an adaptive temporal feature resolution (ATFR) that varies for each input video clip. By integrating the SGS module into current 3D CNNs, we can convert them into more efficient 3D CNNs with ATFR. Our evaluations demonstrate that the proposed module improves upon the state-of-the-art by decreasing computational cost (GFLOPs) by 50% while preserving or even improving accuracy. We evaluate the effectiveness of the SGS module by adding it to multiple state-of-the-art 3D CNNs.

1.2.4 Adaptive Vision Transformers for Efficient Video Classification



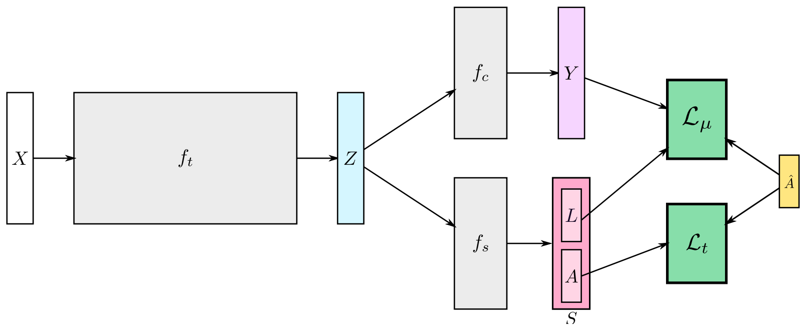
In previous work, we introduced a method for improving the efficiency of 3D CNNs. More recently, vision transformers have demonstrated their effectiveness in image and video classification tasks, following the successes of CNNs in image and video understanding. While state-of-the-art vision transformer models achieve impressive results in image and video classification, they tend to be computationally expensive and have a high computation cost. While the computation cost of a vision transformer can be reduced by decreasing the number of tokens within the network, there is no setting that is optimal for all input images. Therefore, we present a differentiable, parameter-free Adaptive Token Sampler (ATS) module that can be integrated into any existing vision transformer architecture. The ATS module enhances vision transformers by scoring and adaptively sampling significant tokens, resulting in an adaptive number of tokens that varies for each input image or video. By incorporating the ATS module into current transformer blocks, we can convert them into more efficient vision transformers with an adaptive number of tokens. The ATS module is parameter-free, making it easy to add to off-the-shelf, pre-trained vision transformers as a plug-and-play module, reducing their computation cost without any additional training. Additionally, due to its differentiable design, a vision transformer equipped with ATS can be trained. We evaluate the efficiency of the ATS module in both image and video classification tasks by adding it to multiple state-of-the-art vision transformers. Our proposed module improves upon the state-of-the-art by reducing their computational costs by a factor of 2 while maintaining accuracy.

1.2.5 Weakly Supervised Temporal Action Segmentation from Action Sets



Despite the numerous methods that have achieved state-of-the-art results in video understanding on trimmed video clips, real-world scenarios often involve untrimmed videos with multiple actions requiring recognition at each frame. This task, referred to as temporal action segmentation, involves not only recognizing actions within untrimmed videos but also determining their order and duration. While there has been a surge of interest in temporal action segmentation, annotating each frame of a video is a tedious and costly process. As a result, weakly supervised approaches have been developed to learn temporal action segmentation from videos that are only weakly labeled. In such cases, the set of actions present in a video is obtained as weak supervision, providing a cost-effective means of annotating a video. Given a set of actions in which only the list of actions occurring in the video is known, but not when, how often, or in which order they occur, we propose an end-to-end trainable approach to train a temporal action segmentation model with such weak supervision. Our approach divides the video into smaller temporal regions and predicts for each region the action label and its length, as well as estimating action labels for each frame. By measuring the consistency of the frame-wise predictions with respect to the temporal regions and annotated action labels, the network learns to divide the video into class-consistent regions. We evaluate our approach on two datasets and achieve superior results over the preceding methods.

1.2.6 Weakly Supervised Temporal Action Segmentation from Action Transcripts



In spite of the cost-effectiveness of utilizing a set of actions as a means of annotating videos, using it as proper supervision for temporal action segmentation models presents a complicated challenge. As a result, approaches that utilize only a set of actions as supervision are still in the early stages of development in terms of performance. Given the limitations of approaches with weak supervision, methods that can leverage stronger forms of supervision are of great importance. One such type of weak supervision is transcripts, which are ordered lists of actions indicating the order in which actions occur in a training video, but not when they occur. As a culminating contribution, we propose a novel end-to-end framework for weakly supervised action segmentation using a two-branch neural network. The network's dual branches independently predict redundant yet distinct action segmentation representations, and we incorporate a mutual consistency loss term, denoted as MuCon, to enforce consistency between these redundant representations. By integrating the MuCon loss with a loss function for transcript prediction, our approach achieves comparable accuracy to previous methods while demonstrating significant improvements in efficiency, exhibiting a 14-fold decrease in training time and a 20-fold increase in inference speed. Moreover, the effectiveness of the MuCon loss is also showcased in a fully supervised setting.

1.3 Thesis Structure

The rest of this thesis is organized as follows:

Chapter 2 provides a concise overview of the key concepts that are central to the present thesis. It starts with a brief introduction to various types of neural networks, including convolutional, recurrent, and transformer networks. It then describes the video representations that serve as inputs to our proposed temporal segmentation approaches. Following this, it presents a formal definition of the evaluation metrics employed to assess the performance of the proposed approaches. Finally, it presents a summary of the datasets used to evaluate the proposed approaches.

Chapter 3 provides an overview of the existing literature on action recognition and temporal action segmentation.

Chapter 4 proposes the STC block for 3D CNNs. It also proposes a transfer learning approach for transferring the knowledge of a pretrained 2D CNN to a 3D CNN. This chapter is based on *Diba et al.* (2018).

Chapter 5 introduces the novel benchmark of holistic video understanding dataset for the novel task of multi-label video classification. Furthermore, it introduces HATNet, a novel video classification model. This chapter is based on *Diba et al.* (2020).

Chapter 6 proposes the SGS module for 3D CNNs. It shows that by adding SGS to 3D CNNs, we can make them more efficient. This chapter is based on *Fayyaz et al.* (2021).

Chapter 7 extends the idea of adaptive 3D CNNs proposed in Chapter 6, to vision transformers, by introducing the Adaptive Token Sampling module (ATS). It shows that by adding ATS to vision transformers, we can make them more efficient. This chapter is based on *Fayyaz et al.* (2022).

Chapter 8 proposes an approach for temporal action segmentation with extremely weak supervision of actions sets. This chapter is based on *Fayyaz and Gall* (2020).

Chapter 9 proposes the MuCon approach for temporal action segmentation with weak supervision of actions transcripts. This chapter is based on *Souri et al.* (2022).

Finally, conclusions are given in **Chapter 10** along with suggested directions for future work.

CHAPTER 2

Background

In this chapter, we provide an overview of the foundational concepts that are central to the research presented in this thesis. Specifically, we begin by introducing the various types of neural networks that serve as the basis for our proposed methods. Following this, we provide a succinct overview of video representations, highlighting their importance in the context of our work. We then outline the evaluation metrics employed in our experiments, highlighting their appropriateness for evaluating the efficacy of our approaches. Finally, we introduce the datasets utilized in this study, highlighting their relevance to our research goals.

Contents

2.1 Neural Networks

Artificial neural networks (ANNs), also known simply as neural networks, are a prevalent class of machine learning models that have been widely employed in various tasks. Inspired by the architecture of biological neural networks, these models can be represented as a function $g_\theta(\mathcal{X})$ that maps an input $\mathcal{X}$ to an output $\mathcal{Y}$ via learnable parameters θ. In essence, neural networks are learnable models that can approximate functions to a certain degree. A basic form of neural network is the feed-forward neural network, which consists of interconnected layers of processing units known as neurons. Each neuron applies a transformation to the output of the previous layer, with the final output of the network representing the result of the function approximation.

In the following, we provide a brief overview of the various components that make up a neural network, as well as a description of the training procedure utilized to optimize the network's performance.

2.1.1 Neuron

A neuron serves as the fundamental processing unit of a neural network, computing a weighted sum of its inputs and applying a bias to produce an output. This output is then transformed via an activation function, as follows

$$a = f(b + \sum_{i=1}^{K} w_i x_i), \tag{2.1}$$

where a represents the activation, w_j is the weight, x_i is the input, K is the total number of inputs, and $f(.)$ is the activation function. In a feed-forward network, the activations of the neurons in a given layer serve as the inputs to the neurons in the subsequent layer. Figure 2.1 illustrates a neuron. The simplest form of neural network is the perceptron, which consists of a single neuron.

2.1.2 Activation Function

In artificial neural networks, the activation function of a neuron serves to transform the output of that neuron's given set of inputs, often leading to the activation or deactivation of the neuron. To model more complex tasks, it is essential to employ a non-linear activation function, as a deep network with many layers but without non-linear activation function would be equivalent to a single-layer network. There exists a range of activation functions that can be utilized in neural networks, with the linear function serving as one of the simplest. This function maps a real number x to itself, as described below

$$f_{linear}(x) = x. \tag{2.2}$$

The use of a linear function is often reserved for the activation of the final output neuron in a regression model.

Another widely utilized activation function is the sigmoid function, which is characterized by an S-shaped curve in mathematics. There exist various sigmoid functions, with the logistic sigmoid function, described in Equation 2.3, being a popular choice in neural networks.

$$f_{sigmoid}(x) = \frac{1}{1+e^{-x}}. \tag{2.3}$$

This function, often referred to simply as the sigmoid function in machine learning, maps any given real value to the range $(0, 1)$. As such, it is often employed in neural network architectures for modeling probabilities on top of the final model outputs.

The sigmoid function was once a popular choice for use as an activation function in neural networks. However, due to its characteristics, it vanishes the gradients while optimizing the neural network using back-propagation. Therefore, sigmoid functions cannot be used in deep neural networks. To alleviate this issue, Rectified Linear Unit (ReLU) has been introduced. This function defined in Equation 2.4 is a piece-wise linear function that maps a real value x to the maximum of x and 0.

$$f_{ReLU}(x) = max(x, 0) \tag{2.4}$$

The derivative of the ReLU function is constant and equal to 1 for the positive range of its domain, leading to a low computation cost for training deep neural networks equipped with ReLU activation functions. Additionally, the constant derivative of the ReLU function helps to mitigate the vanishing gradient issue of sigmoid activation functions. As a result, ReLU has become a popular activation function in modern deep neural networks.

2.1.3 Network Training

As mentioned before, a neural network is a function that maps the inputs to the desired outputs using some learnable parameters. The parameters of the neural network, are learned through an optimization process over the given training data to meet the objective of the neural network. The objective of a neural network is defined as an objective function that is minimized during the optimization process. A common set of algorithms for optimizing neural networks are based on gradient descent. In a gradient decent-based algorithm, given a set of initial network parameters, the gradients of the objective function with respect to the parameters are computed. The parameters are then iteratively updated in the opposite direction of the gradients

$$\theta_{i+1} = \theta_i - \eta \nabla \mathcal{L}(\theta_i), \tag{2.5}$$

where θ_i are the parameters of the network at the $i - th$ iteration, $\mathcal{L}$ is the objective function, and $\eta \in \mathbb{R}^+$ is the learning rate that controls the updating rate of the parameters. The computation of gradients for the parameters of a deep neural network is a challenging problem. The backpropagation algorithm is used for computing the gradients. The backpropagation algorithm (*Rumelhart et al.*, 1986) uses chain rule to recursively compute the gradients of the parameters at each layer based on the gradients at the following layer.

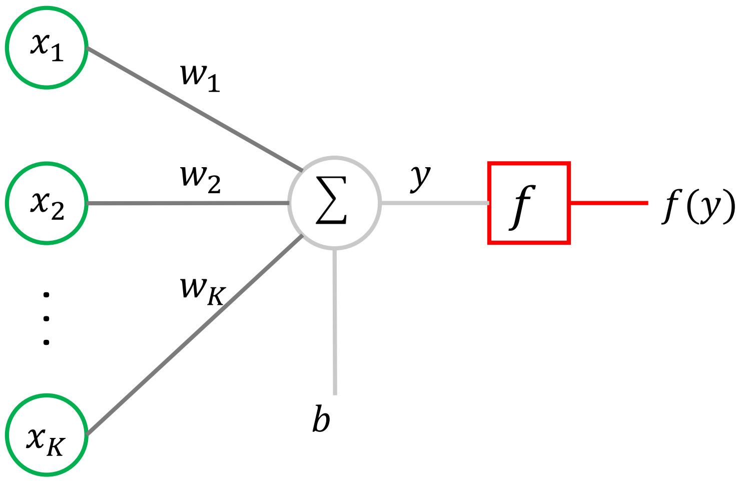


Figure 2.1: A neuron composed of K weights $(w_1, \dots, w_K)$, inputs $(x_1, \dots, x_K)$, a bias (b) and an activation function f. The weights, inputs, and bias term are used to compute a linear combination of the inputs, which is then passed through the activation function to produce the neuron's output. The activation function is typically chosen to introduce non-linearity into the network, allowing it to learn complex mappings between inputs and outputs.

2.1.4 Objective Function

The objective function quantitatively differentiates between the outputs of a neural network and the desired training data. In other words, an objective function can quantitatively measure the performance of a neural network. Therefore, it is also called a loss function. For a multi-class classification task where the outputs of the network are one-hot vectors, the cross-entropy loss is used

$$\mathcal{L}_{ce}(\theta) = -\frac{1}{N}\sum_{n=1}^{N} log(y_c(x_n, \theta)), \tag{2.6}$$

where N is the number of classes, $y_c(x_n, \theta)$ is the output of the network for the target class c.

2.2 Multi Layer Perceptron (MLP)

Multi-layer perception is one of the earliest forms of artificial neural networks. MLP is a fully connected neural network with multiple layers. As shown in Fig.2.2, MLP has 3 different types of layers. The input layer consists of many neurons that are connected to all of the neurons of the following layer. The input data to the network is fed to the input layer. The activations of the input layer are fed to the following hidden layer and similarly, the activations of the hidden layer are fed to the output layer. This process of forwarding data to the following layers is called forward propagation. The output activations of the output layer are indeed the outputs of the neural network.

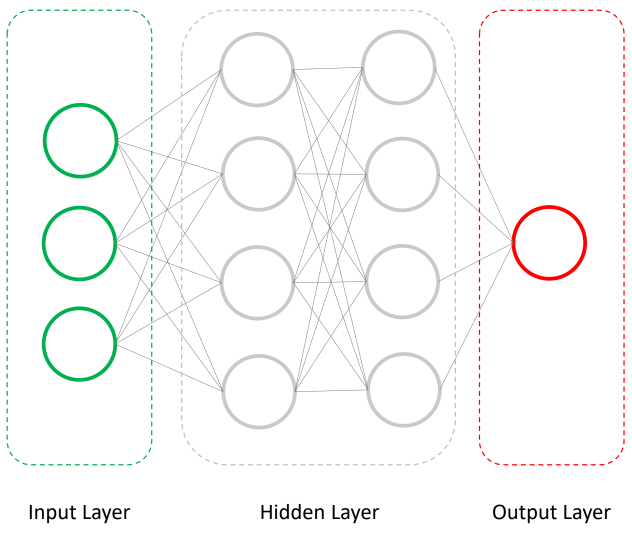


Figure 2.2: A feed-forward artificial neural network composed of two intermediary layers, each comprising a multitude of perceptrons, with each perceptron being fully connected to the entirety of the perceptrons in the preceding layer. This architecture is commonly used in various computer vision tasks such as object classification and detection, due to its ability to extract high-level features from the input data through the non-linear activation functions in the hidden layers.

2.3 Recurrent Neural Networks (RNNs)

As previously mentioned, feedforward neural networks are a class of artificial neural networks characterized by connections between neurons in different layers, but not within the same layer. These networks are highly effective at approximating functions, but are not equipped to represent functions with temporal components. In contrast, recurrent neural networks (RNNs) feature connections that form a directed cycle, allowing them to retain information from previous input and output sequences. This property makes RNNs particularly useful for tasks such as language modeling, time series prediction, and video understanding. For an input sequence $X_{1:T} = (x_1, \dots, x_T)$, the RNN updates its state h_t at step t based on the previous state h_{t-1} and the current input x_t. A vanilla RNN combines h_{t-1} and x_t linearly using the following equation:

$$h_t = f(W_h h_{t-1} + W_x x_t + b), \tag{2.7}$$

where W_h and W_x are the weight matrices that determine the influence of the previous state and the current input on the current state, respectively, and b is the bias term. The function f is a non-linear activation function, that is used to introduce non-linearity into the model. This allows the RNN to learn more complex patterns in the data. Figure 2.3 depicts a vanilla RNN. The parameters W_h, W_x, and b are shared across all time steps and are applied recursively to the inputs from the sequence. This allows RNNs to handle input sequences of arbitrary length. To train RNNs, the back-propagation through time (BPTT) algorithm is commonly used. This involves first unfolding the network in time, using the same parameters at each time step, and then using the back-propagation

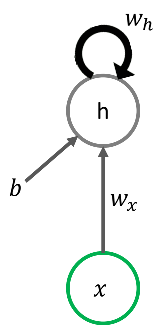


Figure 2.3: A vanilla recurrent neural network (RNN) architecture, wherein the hidden state h_t at each time step t is a function of both the state from the previous time step h_{t-1} and the input x_t at the current step. The parameters W_x, W_h, and b, which govern the dynamics of the hidden state, are shared over time.

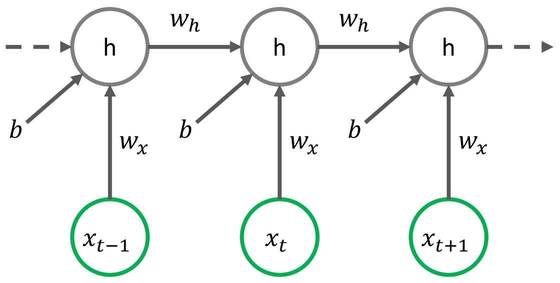


Figure 2.4: A vanilla recurrent neural network (RNN) architecture, unfolded in time, wherein the hidden state h_t at each time step t is a function of both the state from the previous time step h_{t-1} and the input x_t at the current step. The parameters W_x, W_h, and b, which govern the dynamics of the hidden state, are shared over time.

algorithm to update the network parameters. Figure 2.4 illustrates an unfolded vanilla RNN through time. The use of gradient descent-based training methods for RNNs can suffer from problems such as vanishing or exploding gradients (*Bengio et al.*, 1994), which can limit the performance of the network on long-term dependencies and complex sequential data. These problems arise due to the backpropagation of error gradients through time, which can result in gradients that either vanish to zero or explode to large values, making it difficult for the network to learn effectively. To address this issue, several RNN variants, such as long short-term memory (LSTM) (*Hochreiter and Schmidhuber*, 1997) networks and gated recurrent units (GRUs) (*Cho et al.*, 2014a), have been developed that incorporate multiplicative gates in the RNN update rule. These gates act as trainable "forget" and "input" filters, allowing the network to selectively retain or discard information from previous time steps. This helps to prevent the error gradients from vanishing, allowing the network to learn more effectively from long sequences.

2.3.1 Long Short-Term Memory (LSTM)

Long short-term memory (LSTM) networks are a type of recurrent neural network (RNN) that was introduced by *Hochreiter and Schmidhuber* (1997) to overcome the limitations of traditional RNNs. LSTM networks are equipped with a memory cell that is updated by multiplicative gates, enabling them to selectively retain or discard information from previous time steps. This makes LSTM networks more effective and stable when dealing with long-term dependencies and complex sequential data. An LSTM cell is a computational unit that takes as input the previous hidden state, the current input, and a set of learned weights. It can be defined as follows. Let x_t be the input at time step t, h_{t-1} be the previous hidden state, and c_{t-1} be the previous cell state. The cell computes the current cell state c_t and hidden state h_t using the following equations:

$$\begin{aligned}
i_t &= \sigma(W_{xi}x_t + W_{hi}h_{t-1} + b_i), \\
f_t &= \sigma(W_{xf}x_t + W_{hf}h_{t-1} + W_{cf}c_{t-1} + b_f), \\
c_t &= f_t \odot c_{t-1} + i_t \odot \tanh(W_{xc}x_t + W_{hc}h_{t-1} + b_c), \\
o_t &= \sigma(W_{xo}x_t + W_{ho}h_{t-1} + W_{co}c_t + b_o), \\
h_t &= o_t \tanh(c_t),
\end{aligned} \tag{2.8}$$

where, W_{*i}, W_{*f}, W_{*c}, and W_{*o} are the weight matrices for the input gate, forget gate, cell state, and output gate, respectively, and b_i, b_f, b_c, and b_o are the corresponding biases, σ is the sigmoid function, and i_t, f_t, and o_t are the input, forget, and output gate, respectively.

The forget gate of an LSTM unit plays a critical role in determining the evolution of the memory cell at each time step. By assessing the relevance of the information stored in the memory cell, the forget gate determines what should be discarded, allowing the LSTM to selectively retain only the most pertinent information. Meanwhile, the input gate assesses the incoming data and decides what should be incorporated into the updated memory cell. Finally, the output is computed by multiplying the contents of the memory cell by the output gate, which serves as a filter to produce the final output of the LSTM unit. This mechanism allows the LSTM to selectively retain and access relevant information, enabling it to effectively model long-term dependencies in sequential data.

2.4 Convolutional Neural Networks (CNNs)

Convolutional neural networks (CNNs) are a type of neural networks that has been successfully applied to a wide range of tasks, including image and video classification, and object detection. CNNs are composed of convolutional layers, which are designed to learn spatial hierarchies of features in the input data. In contrast to traditional neural networks, which use fully-connected layers where each neuron receives connections from all neurons in the previous layer, convolutional layers exhibit sparse connectivity, where each neuron is connected to a local region of the previous layer. This allows the network to be translation invariant, meaning that it is robust to small translations and deformations of the input. Additionally, the use of shared weights across the spatial dimensions of the input reduces the number of parameters in the model and improves its generalization, leading to better performance on unseen data.

Convolutional layers use a fixed-size kernel, also known as a filter, to scan the input and compute a set of feature maps. The kernel is convolved with the input, element-wise multiplying and summing the values in the input region defined by the kernel. The resulting output is passed through a nonlinear activation function, such as a rectified linear unit (ReLU), to introduce nonlinearities and improve the expressiveness of the model. The use of convolutional layers allows CNNs to have several desirable properties. First, the learned filters can be interpreted as local feature detectors, such as edges, corners, and textures, which can be composed to form more complex structures. This provides a compact representation of the input. Second, the use of shared weights across the spatial dimensions of the input reduces the number of parameters in the model and improves generalization, leading to better performance on unseen data. Convolutional layers are characterized by the use of shared weights across the spatial dimensions of the input. This allows the network to learn multiple kernels, or filters, using a relatively small number of parameters. The convolutional layer is then simply a convolution operation between the input image I and the learned kernel W. Formally, a two-dimensional convolution is defined as follows:

$$(I * W)(i, j) = \sum_m \sum_n I(i - m, j - n) \cdot W(m, n), \tag{2.9}$$

here, $(I * W)(i, j)$ denotes the output of the convolution at location (i, j) in the input, and $I(i - m, j - n)$ and $W(m, n)$ denote the values of the input and kernel at spatial offsets (m, n) relative to the origin of the receptive field. This operation allows the network to learn local patterns in the input data, which can be composed to form more complex structures. The use of shared weights and the convolution operation enable convolutional layers to be highly efficient in terms of both computation and the number of parameters, making them a key component of convolutional neural networks (CNNs).

In addition to convolutional layers, CNNs typically include pooling layers, which are used to downsample the spatial dimensions of the input. This helps reduce the computational complexity of the model and improves its ability to learn spatial hierarchies. It also helps make the model invariant to small translations and deformations of the input. The output of the convolutional and pooling layers is typically fed into one or more fully-connected layers, which perform the final classification or regression task. Overall, the use of convolutional layers enables CNNs to learn powerful feature representations of the input, leading to their success in various applications.

There have been many successful convolutional neural network architectures. One of the earliest 2D CNNs is LeNet-5 (*Lecun et al.*, 1998), which was applied to recognize handwritten digits. Other popular and widely used architectures include AlexNet (*Krizhevsky et al.*, 2012) and VGG (*Simonyan and Zisserman*, 2015), which have been applied to tasks such as image classification and object detection. More recently, many architectures are based on ResNet (*He et al.*, 2016b), which has proven to be highly effective for various tasks. Overall, the continued development of CNN architectures has enabled significant progress in the field of machine learning, with many successful applications in diverse domains.

2.5 Spatio-Temporal Convolutional Neural Networks (3D-CNNs)

Spatio-temporal convolutional neural networks (3D-CNNs) have garnered significant attention in the realm of video analysis, specifically for tasks such as action recognition and object tracking. These networks diverge from traditional convolutional neural networks (CNNs) in their aptitude to extract both spatial and temporal features from video sequences.

A standard CNN processes each frame independently, extracting spatial features through a series of convolutional layers. Conversely, an 3D-CNN captures both spatial and temporal information by applying 3D convolutions across the spatial and temporal dimensions of the input tensor. The convolutional layer is then simply a convolution operation between the input video V and the learned kernel W. Formally, a three-dimensional convolution is defined as follows:

$$(V * W)(i, j, k) = \sum_m \sum_n \sum_p V(i - m, j - n, k - p) \cdot W(m, n, p), \tag{2.10}$$

where $(V * W)(i, j, k)$ denotes the output of the convolution at location (i, j, k) in the input, and $V(i - m, j - n, k - p)$ and $W(m, n, p)$ denote the values of the input and kernel at spatial offsets (m, n, p) relative to the origin of the receptive field. The utilization of 3D convolutions enables 3D-CNNs to capture both spatial and temporal dependencies in the input video, allowing them to effectively model dynamic processes such as human actions.

Spatio-temporal convolutional neural networks (3D-CNNs) have garnered widespread use for a variety of video analysis tasks due to their ability to extract both spatial and temporal features from the input data. One of the earliest 3D CNNs, C3D (*Tran et al.*, 2015), was used for the classification of human action videos and comprises a series of 3D convolutional and pooling layers followed by fully-connected layers. Other notable 3D-CNN architectures include I3D (*Carreira and Zisserman*, 2017), which combines 3D convolutions with different kernels for enhanced performance, and R(2+1)D (*Tran et al.*, 2018b), which employs 2D convolutions in the spatial dimensions and 1D convolutions in the temporal dimension.

Despite the challenges associated with the implementation of spatio-temporal convolutional neural networks (3D-CNNs), such as the requirement for a significant amount of annotated training data and the computational intensity of training and deployment, these networks have demonstrated their effectiveness in video analysis tasks. In this thesis, we propose the incorporation of novel modules, architectures, and datasets in an effort to enhance the capabilities of 3D-CNNs. Despite the challenges inherent to 3D-CNNs, their efficacy as a formidable tool for video analysis tasks has been demonstrated.

2.6 Temporal Convolutional Networks (TCNs)

In contrast to spatio-temporal convolutional neural networks (3D-CNNs), temporal convolutional neural networks (TCNs) utilize temporal convolutional layers that exclusively capture temporal information. TCNs are particularly well-suited for tasks involving sequential data, such as video understanding, due to their ability to capture long-range dependencies in the input data.

TCNs, as a specialized variant of traditional CNNs, have the ability to incorporate pooling layers and utilize CNN-related concepts such as padding, strided convolution, and dilated convolutions. Specifically, dilated convolutions with dilation factor d involve the insertion of $d - 1$ zeros between consecutive kernel elements, resulting in an expanded kernel capable of effectively extracting temporal features from sequential data. The ability of TCNs to adapt these techniques from CNNs makes them a valuable asset in video understanding tasks. In 2019, *Abu Farha and Gall* (2019) introduced MS-TCN, a temporal convolutional neural network that demonstrated exceptional capabilities in temporal video segmentation. Building upon the success of *van den Oord et al.* (2016), this work also employs dilated temporal convolutional neural networks for temporal video segmentation tasks. Formally, the one-dimensional dilated temporal convolution is defined as follows:

$$(X * W)(i) = \sum_j X(i - jd) \cdot W(j), \tag{2.11}$$

where $(X * W)(i)$ denotes the output of the convolution at position i in the input sequence, and $X(i - jd)$ and $W(j)$ denote the values of the input and kernel at temporal offsets j relative to the origin of the receptive field, with a dilation factor of d.

Temporal convolutional networks (TCNs) present several advantages over recurrent neural networks (RNNs) for tasks involving sequential data. Unlike RNNs, TCNs are able to handle input sequences of arbitrary length and do not suffer from the vanishing and exploding gradients problems that can occur in RNNs when the sequence is lengthy. This makes TCNs a suitable choice for tasks such as language translation, speech recognition, and video understanding that involve long sequences. In addition, TCNs can be trained using standard backpropagation techniques, without the need for specialized algorithms such as truncated backpropagation through time (BPTT) or teacher forcing, which are required for training RNNs. This simplifies the training process and makes it easier to implement TCNs in practice. Furthermore, the output of TCNs at time t is dependent solely on the input sequence and does not incorporate the output at previous time steps, enabling the parallel computation of the output at all time steps and significantly increasing the efficiency of TCNs compared to RNNs, particularly for long sequences. The parallelizability of TCNs is achieved through the use of temporal convolutional layers, which capture long-range dependencies in the input data through the sharing of kernel weights across time steps. This allows TCNs to model temporal dependencies without the need for recurrence, making them more efficient and easier to parallelize than RNNs.

Temporal convolutional networks (TCNs) offer a number of benefits over recurrent neural networks (RNNs) for tasks involving sequential data, including their capacity to process input sequences of arbitrary length, the absence of vanishing and exploding gradients problems, and the ability to be trained using standard backpropagation techniques. Despite these advantages, TCNs may require a significant amount of annotated training data and can be computationally intensive to train and deploy. To address this issue, this thesis introduces methods for training TCNs on video data with weak annotations.

2.7 Vision Transformers

The Transformer neural network architecture, introduced in 2017 (*Vaswani et al.*, 2017), has proven to be a highly innovative and influential approach for processing sequential data. Its core is composed of a series of self-attention blocks, which allow for efficient and effective processing through the use of dot-product attention mechanisms. These mechanisms enable the network to compute the relevance of each element in the input sequence to every other element, producing attention weights that are utilized to compute a weighted sum of value vectors representing the input elements. This process is repeated for each head in the multi-headed attention mechanism, with the outputs being concatenated and transformed to generate the final self-attention output. In addition to self-attention mechanisms, the Transformer architecture also incorporates feed-forward neural network layers, implemented using multi-layer perceptrons, to further process the self-attention output. These layers are designed to learn more complex relationships within the data. The integration of self-attention mechanisms and feed-forward layers enables the Transformer network to effectively capture both short-range and long-range dependencies within sequential data.

Following the successful implementation of the Transformer neural network architecture in natural language processing, Vision Transformers (*Dosovitskiy et al.*, 2021) have gained widespread usage in the field of computer vision. The emergence of the Vision Transformer (ViT) marked a significant shift in the field of computer vision, as it presented a competitive alternative to the currently state-of-the-art convolutional neural networks (CNNs) that dominate the domain of image recognition tasks. At its core, ViT utilizes a transformer architecture, which consists of a series of self-attention blocks that are used to process sequential data in an efficient and effective manner. In the context of ViT, the input image is first divided into a set of non-overlapping patches, which are then fed to feed-forward neural networks to be represented as patch embeddings of lower dimensionality. These vector embeddings, or tokens, are then merged with positional encodings, which serve to preserve the relative spatial relationships between the patches in the original image. Positional encodings are typically implemented through the use of sinusoidal functions, which are added to the embedding vectors. The resulting embeddings are then concatenated with an auxiliary token called the classification token, and fed to the first transformer block. Each transformer block in a ViT consists of multiple linear layers for encoding the input tokens to keys, queries, and values. These are then fed to the multi-headed self-attention mechanism. Self-attention mechanisms are implemented through the use of dot-product attention, which allows the network to compute the relevance of each element in the input sequence to every other element. This is achieved through the use of dot-product attention equations, which compute the dot-product between the query $\mathcal{Q}$ and key $\mathcal{K}$ vectors for each element in the sequence:

$$\text{Attention}(\mathcal{Q}, \mathcal{K}, \mathcal{V}) = \text{softmax}\left(\frac{\mathcal{Q}\mathcal{K}^\top}{\sqrt{d_k}}\right)\mathcal{V} \tag{2.12}$$

where $\mathcal{V}$ represents the value vectors, which represent the input elements, and d_k is the dimensionality of the key vectors. The resulting attention weights are then used to compute a weighted sum of the value vectors, which is the self-attention output. This process is repeated for each head in the multi-headed attention mechanism, with the outputs being concatenated and transformed to produce the final self-attention output. The output of the self-attention mechanism is then fed to a fully connected network, which serves to further process the self-attention output and provide the final output of the transformer block. In a ViT, multiple transformer blocks are typically stacked on top of each other, with the output of one block serving as the input to the next. This allows the network to capture increasingly complex relationships within the data as it progresses through the transformer blocks. Finally, the final classification output token from the final transformer block of the ViT is fed to a classification head, which is a fully connected network. The class probabilities are then calculated via a softmax operation on top of the output logits of the classification head. This allows the network to output a probability distribution over the different classes in the dataset, indicating the likelihood that the input image belongs to each class.

Vision transformers are highly effective neural network architectures for image classification tasks due to their ability to capture increasingly complex relationships through multiple transformer blocks. However, their computational intensity can pose challenges during training and deployment. In this work, we introduce a method to make vision transformers more efficient in order to address this issue.

2.8 Video Vision Transformers

Building upon the successes of Vision Transformer in image classification, TimeSformer (*Bertasius et al.*, 2021) proposed a transformer-based architecture for video understanding by extending the self-attention mechanism of standard transformer models to video data. As previously mentioned, a ViT takes as input a set of tokens that represent 3-dimensional patches ($p \in \mathbb{R}^{W \times H \times C}$) of an input image, where W, H, and C represent the width, height, and number of channels of the patches, respectively. To process a video, the video can be divided into 4-dimensional patches ($p \in \mathbb{R}^{T' \times W \times H \times C}$), where T' is the temporal dimension. These patches can then be represented as input tokens $\mathcal{I} \in \mathbb{R}^{T \times S}$, where T is the number of temporal locations and S is the number of spatial locations. In transformer models, self-attention allows each element in a sequence to attend to all other elements in the sequence, enabling the network to capture relationships between elements that are far apart in the sequence. This is in contrast to convolutional neural networks, which are limited to capturing relationships between nearby elements. In the context of video understanding, the self-attention mechanism is applied to both temporal and spatial locations in the video data. However, the application of self-attention to both temporal and spatial locations in video data can result in a computational complexity of $O(T^2S^2)$, which may be infeasible for long video sequences or high-resolution video data. To address this issue, TimeSformer introduced a complexity of $O(T^2S + TS^2)$ for the self-attention mechanism, which although still computationally intensive, significantly reduces the complexity of the original self-attention mechanism. In an effort to further improve the efficiency of video transformer models, X-ViT (*Bulat et al.*, 2021) proposed an efficient video transformer that reduces the complexity of the self-attention mechanism to $O(TS^2)$. This represents a significant advancement in the field of video understanding, as it allows for the application of transformer-based models to a wider range of video data.

Despite these efforts to make video vision transformers more efficient, they are still expensive to train and deploy. In this thesis, we propose a method that not only makes image classification vision transformers more efficient, but also makes video vision transformers more efficient.

2.9 Video Representation

Video data is typically represented as a 4-dimensional tensor *i.e.* $v \in \mathbb{R}^{T \times W \times H \times C}$, with dimensions corresponding to the number of frames T, spatial resolution $W \times H$, and color channels C. While this raw video data is often fed directly into deep learning models for action recognition task, it is not feasible to do so for tasks such as temporal action segmentation due to the large number of frames in some videos. To overcome this issue, video frames are usually preprocessed using a feature extraction algorithm, resulting in an D-dimensional vector representation for each frame. This results in a video representation that is a tensor of size $T \times D$, where T is the number of frames, which serves as the input for tasks such as temporal action segmentation.

Traditionally, video representation has relied on hand-crafted features, such as improved dense trajectories (*Wang and Schmid*, 2013), which are designed to capture specific spatial and temporal patterns in the input data. However, these hand-crafted features have limited performance due to their

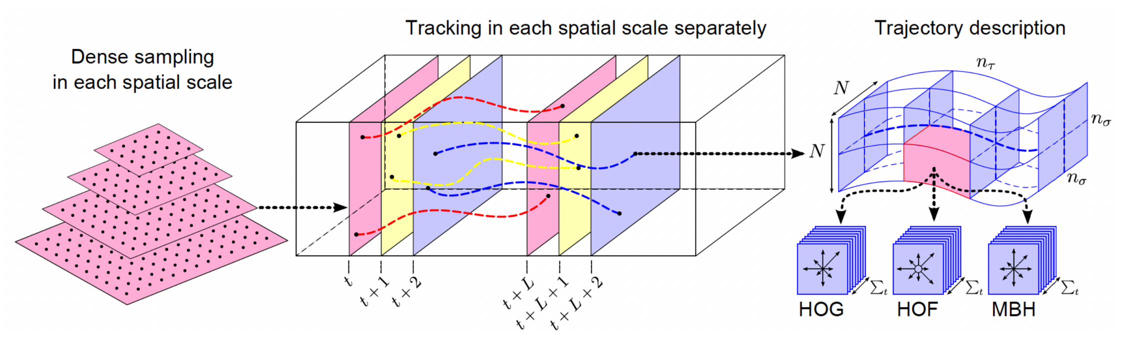


Figure 2.5: A pipeline for extracting dense trajectories from a video is presented. The pipeline comprises of three stages: dense sampling of feature points on a grid for different spatial scales (left), tracking of these points in each spatial scale for L frames based on a dense optical flow field (middle), and computation of histograms of oriented gradients (HOG), histograms of optical flow (HOF), and motion boundary histograms (MBH) along the trajectory in a $N \times N$ pixels neighborhood, which is divided into $n_\sigma \times n_\sigma \times n_\tau$ cells (right). The feature points are densely sampled on a grid, at different spatial scales, to ensure comprehensive coverage of the video frames. These points are then tracked in each scale for L frames, based on a dense optical flow field, to obtain dense trajectories. The histograms of oriented gradients (HOG), histograms of optical flow (HOF), and motion boundary histograms (MBH) are computed along the trajectory in a $N \times N$ pixels neighborhood, which is divided into $n_\sigma \times n_\sigma \times n_\tau$ cells to capture the spatio-temporal information of the trajectories. The figure is taken from *Wang et al.* (2013)

simplicity and lack of generalization ability.

To address these limitations, recent works have proposed the use of deep learned features, which are extracted from convolutional neural networks (CNNs) trained on large-scale video datasets. These deep learned features have demonstrated superior performance on a wide range of video analysis tasks. In this thesis, we consider using two types of feature representations for temporal action segmentation tasks: features based on improved dense trajectories and features extracted using 3D CNNs.

2.9.1 Improved Dense Trajectories (IDT)

Improved Dense Trajectories (IDT) is a hand-crafted approach for video feature representation, initially proposed for action recognition tasks (*Wang and Schmid*, 2013). IDT involves the computation of dense optical flow, which is pre-processed to smooth out noise and compensate for camera motion. Subsequent to this, feature points are densely sampled at multiple scales and tracked using the dense optical flow, resulting in the formation of trajectories. These trajectories are then encoded using histograms of oriented gradients (HOG), histograms of optical flow (HOF), and motion boundary histograms (MBH), providing a rich descriptor for each frame of the video. Figure 2.5 illustrates the the process of extracting IDT features from video.

Kuehne et al. (2016a) extended the use of IDT features to the task of temporal action segmentation, where they are further processed using Principal Component Analysis (PCA) to reduce their dimensionality to 64. Fisher vectors (*Perronnin et al.*, 2010) are then applied to produce the final representation, which is power and l_2 normalized according to the method proposed by *Sánchez et al.*

(2013). This results in a 64-dimensional feature representation for each frame of the input video, yielding a tensor of size $x \in \mathbb{R}^{T \times 64}$, where T is the number of frames in the video.

It is worth noting that IDT, while effective at the time of its proposal, has since been surpassed by more recent approaches based on deep learning. Deep learning methods learn feature representations automatically from the data, often yielding improved performance compared to hand-crafted features such as IDT. Additionally, deep learning approaches are more flexible, as they can learn complex feature hierarchies and adapt to various tasks and datasets. As a result, deep learning features have become the de facto choice for many computer vision tasks. In this study, we employ IDT features, introduced by *Kuehne et al.* (2016a), for temporal action segmentation tasks and compare their performance to deep learning-based features. By doing so, we aim to gain a deeper understanding of the influence of video representations on downstream tasks such as temporal action segmentation.

2.9.2 Deep Learned Features

The efficacy of deep learning approaches in learning feature representations from data in the field of computer vision has been well-established. In particular, it has been demonstrated that the output of the penultimate layer of a deep neural network can serve as a robust feature representation for an input when the weights of the network are fixed (*Sharif Razavian et al.*, 2014). Additionally, the utilization of pre-trained deep neural networks - specifically those trained on the ImageNet dataset (*Deng et al.*, 2009) - can be effectively leveraged as feature extractors for a range of visual recognition tasks. Initially, convolutional neural networks (CNNs) struggled to match the performance of IDT-based features for video classification (*Karpathy et al.*, 2014a). However, the introduction of the two-stream network (*Simonyan and Zisserman*, 2014) marked a turning point, as deep learning models began to achieve performance on par with state-of-the-art hand-crafted features. A two-stream network comprises of two parallel CNNs, including the spatial or appearance stream, which processes RGB frames of videos, and the temporal stream, which processes stacks of optical flow frames. The outputs of these two streams are fused and passed to a classifier. Subsequent methods have built upon this two-stream architecture, resulting in significant improvements. To address the challenges of designing and training 3D CNNs from scratch, *Carreira and Zisserman* (2017) proposed the I3D network, which inflates a successful 2D CNN architecture into 3D and initializes the 3D CNN weights using pre-trained weights from the ImageNet dataset (*Deng et al.*, 2009). As a popular choice for input representation in temporal action segmentation, the I3D network (*Carreira and Zisserman*, 2017) - which has been pre-trained on the kinetics dataset (*Kay et al.*, 2017) for the task of action recognition - has gained widespread adoption (*Abu Farha and Gall*, 2019; *Fayyaz and Gall*, 2020; *Souri et al.*, 2022; *Li et al.*, 2020a; *Yi et al.*, 2021). Specifically, both the RGB and optical flow variants of the I3D network are utilized, with the representations from each modality being concatenated. In this scenario, a video with length T is represented as $v \in \mathbb{R}^{T \times 2048}$, where 2048 is the feature dimension of the final convolutional layer of the I3D network, doubled to accommodate the concatenation of the RGB and optical flow representations. In this thesis, we utilize the I3D model, which has been pre-trained on the Kinetics-400 dataset (*Kay et al.*, 2017), as a means of extracting frame-level features. To generate a feature vector $x_t \in \mathbb{R}^{1 \times 2048}$ for a specific video frame v_t, we process a window of 21 frames centered on that frame through the pre-trained I3D model, including both RGB frames and optical flow frames. The output of the penultimate layer from each stream is then obtained, and the final feature vector for the frame is obtained by concatenating the output from both the appearance and temporal streams. This approach allows us to effectively utilize the pre-trained I3D model for the task of frame-level feature extraction.

2.10 Evaluation Metrics

In the following section, we outline the procedures for calculating evaluation metrics for action recognition, multi-class video classification, and temporal action segmentation. Specifically, we detail the methodologies used to assess the performance of the suggested methods in this thesis.

2.10.1 Accuracy

Accuracy in machine learning refers to the percentage of predictions made by a model that are correct. It is commonly used as a metric for evaluating the performance of action recognition models, and is defined as:

$$Accuracy = \frac{Number\ of\ Correct\ Predictions}{Total\ Number\ of\ Predictions}. \tag{2.13}$$

In simpler terms, accuracy gauges the ability of a model to correctly predict the outcome of a given input. It is a vital consideration in the design and training of machine learning models as high accuracy can translate to improved performance and increased confidence in the results. In this thesis, we utilize this metric to evaluate the action recognition tasks and determine the suitability of the methods for classifying the specified video clips.

2.10.2 Mean Average Precision (mAP)

Mean average precision (mAP) is a commonly used evaluation metric in the field of machine learning, particularly in the context of multi-label classification tasks. It is a measure of the average precision achieved across a set of test examples, and is computed as the mean of the average precision scores of each class in the dataset. In the context of multi-label classification, mAP is used to evaluate the performance of the model on the task of predicting the presence or absence of multiple labels for a given input example. It is a useful metric for evaluating the performance of multi-label classification algorithms, as it provides a comprehensive assessment of the model's performance across all label classes in the dataset. The mAP metric is typically computed using the mean of the average precision scores of each class, as follows:

$$mAP = \frac{1}{C} \sum_{c=1}^{C} AP_c \tag{2.14}$$

where C is the number of classes in the dataset, and AP_c is the average precision score for class c. The average precision score for a particular class is defined as the area under the precision-recall curve for that class. The precision-recall curve is a plot of the precision of the model on the y-axis and the recall on the x-axis, and is obtained by varying the decision threshold of the model. The area under the curve is computed using the trapezoidal rule, and is a measure of the model's ability to achieve high precision at high recall levels. In this thesis mAP is used as an evaluation metric to measure the performance of models on the task of predicting multiple labels for each video in our novel Holistic Video Understanding dataset (HVU). As the HVU dataset consists of videos with multiple labels, mAP provides a comprehensive assessment of the model's performance across all label classes in the dataset.

2.10.3 Mean Over Frame Accuracy (MoF)

The mean over frames accuracy (MoF) is a metric widely used to evaluate the performance of machine learning models on the task of video action segmentation. MoF is defined as the ratio of correctly predicted frames to the total number of frames in the dataset. This metric has been widely adopted in the literature (*Abu Farha and Gall*, 2019; *Kuehne et al.*, 2016a; *Richard et al.*, 2017, 2018b,a). More precisely, MoF is defined as:

$$MoF = \frac{\sum_{i=1}^{V} C_i}{\sum_{i}^{i=V} F_i} \tag{2.15}$$

where V is the number of videos, C_i is the number of correct predicted frames for video i, and F_i is the number of frames of the video i. In this thesis we use MoF to evaluate the temporal action segmentation tasks.

2.10.4 Matching Score

The matching score is another metric that is widely used to evaluate the performance of machine learning models on the task of video action segmentation. The matching score (*Lambert*, 2019) measures the similarity of the predicted transcript compared to the ground truth transcript:

$$\text{Matching Score} = 2 \times \frac{\text{number of matches}}{|A| + |\hat{A}|} \tag{2.16}$$

where $|A|$ is the length of the predicted transcript and $|\hat{A}|$ is the length of the ground truth transcript.

2.10.5 Intersection Over Detection (IoD)

Another metric that is widely used for the task of action segmentation is the Intersection Over Detection (IoD). IoD is only defined for the alignment task since the transcript and number of segments are known for this task. This provides a one to one matching between the predicted and ground truth segments. The IoD is then computed by the average intersection of a ground truth segment with an associated predicted segment divided by the length of the predicted segment, i.e., $|S \cap \hat{S}|/|S|$. In this thesis, we adopt the same definition and code as that employed by *Ding and Xu* (2018) and *Li et al.* (2019).

2.10.6 F1 Score

The F1 Score bears a resemblance to the matching score in that it is computed over segments while also taking into account the temporal position of predicted and ground truth segments. However, it remains unaffected by minor deviations in temporal boundaries. The IoU for a given predicted segment and ground truth segment is determined as

$$IoU = \frac{predictedsegment \cap groundtruthsegment}{predictedsegment \cup groundtruthsegment}. \tag{2.17}$$

A mapping is initially established between ground truth segments and predicted segments based on their respective IoUs. Each ground truth segment is assigned to the predicted segment with the highest IoU. If a predicted segment shares the same label as the corresponding ground truth segment and the IoU ratio exceeds k, it is classified as a true positive (TP). Otherwise, it is deemed a false positive (FP). When a ground truth segment lacks a mapped predicted segment, it is considered a false negative (FN).

To compute the F1 Score for a given threshold of k, precision and recall are first determined using

$$Precision = \frac{TP}{TP + FP}, \tag{2.18}$$

$$Recall = \frac{TP}{TP + FN}. \tag{2.19}$$

Finally, F1 score is subsequently calculated via

$$F1@k = 2.\frac{Precision.Recall}{Precision + Recall}. \tag{2.20}$$

2.11 Datasets

In this section, we outline the datasets utilized in our study. These datasets have been selected for their relevance and representativeness in the domain of our investigation, and have played a crucial role in informing our analysis and evaluation of the proposed methodologies. We provide a brief overview of each dataset, highlighting any pertinent characteristics or features that are relevant to the current work.

2.11.1 UCF101

UCF-101 (*Soomro et al.*, 2012) is a widely used benchmark dataset for the task of action recognition, containing a total of 13,000 videos divided into 101 action classes. The dataset is traditionally split into three subsets, each containing approximately 9,500 videos, and the reported accuracy for the UCF-101 dataset is typically calculated as the average performance across all three splits. The UCF-101 dataset has played a crucial role in the development and evaluation of various action recognition methods.

2.11.2 HMDB51

The HMDB-51 (*Kuehne et al.*, 2011) dataset is a benchmark dataset commonly used for the task of action recognition, containing approximately 7,000 videos divided into 51 action classes. The dataset is divided into three splits for the purposes of training and validation, and the reported accuracy is calculated as the average performance across all three splits. Similar to the UCF-101 dataset, the HMDB-51 dataset has been widely utilized in the development and evaluation of various action recognition methods and serves as a valuable resource for researchers in the field.

2.11.3 Kinetics

The Kinetics dataset (*Kay et al.*, 2017) is a seminal resource within the domain of computer vision, specifically in the realm of action recognition. Comprised of over 400/600/700 action categories, the Kinetics dataset presents a diverse yet formidable dataset for the development and assessment of action recognition models. The videos contained within the Kinetics dataset were sourced from YouTube, with an average duration of 10 seconds, and showcase a wide range of sources, lighting conditions, backgrounds, and camera angles, providing a realistic representation of real-world scenarios. These video clips depict a broad spectrum of human-object interactions, such as playing instruments, as well as human-human interactions, such as shaking hands and hugging. Each video clip is human-annotated with a single action class. The Kinetics dataset has played a crucial role in the advancement of action recognition research, with the I3D network (*Carreira and Zisserman*, 2017) being pre-trained on the dataset and achieving enhanced performance on various action recognition tasks. The Kinetics dataset continues to be extensively utilized within the research community and is a pivotal resource for the development of state-of-the-art action recognition models.

2.11.4 Breakfast

The Breakfast dataset (*Kuehne et al.*, 2014) is a comprehensive collection of videos showcasing various cooking activities, specifically those related to the preparation of breakfast dishes. Comprised of over 1.7k videos, this dataset spans 10 distinct categories of breakfast preparation, including preparing cereal and coffee, each of which comprises 48 finer-grained actions, including background. These activities were carried out by 52 different subjects and captured in 18 different kitchens. The activities depicted in the Breakfast dataset are composed of sequences of more fine-grained actions, such as taking a bowl and pouring cereals. The Breakfast dataset was captured using multiple synchronized cameras in a multi-view setting, which allowed for cost-effective annotation through the synchronization of recordings. The average duration of the videos in this dataset is 2.3 minutes, with the longest video exceeding 10 minutes and the shortest video just over 12 seconds. On average, each video contains 6.9 segments. In our temporal action segmentation experiments, we utilized the 4 train/test splits provided with the Breakfast dataset and report the mean results. This dataset is a valuable resource for the task of temporal action segmentation, due to the rich temporal structure of the videos and the availability of detailed annotations.

2.11.5 Hollywood Extended

The Hollywood extended dataset (*Bojanowski et al.*, 2014) is a collection of 937 video sequences drawn from Hollywood films, depicting 16 diverse action classes such as answering the phone and driving a car. Comprised of roughly 800,000 frames, this dataset exhibits a comparatively high proportion of background frames, with approximately 61% of the frames classified as such. This characteristic distinguishes the Hollywood extended dataset from numerous other datasets. In contrast to the Breakfast dataset (*Kuehne et al.*, 2014), the non-background actions in this dataset are relatively scarce. In our experiments, we adopt the train/test split strategy utilized in previous research (*Richard et al.*, 2017, 2018b,a).

CHAPTER 3

Related Work

In this chapter, we provide a literature review of the action recognition and temporal segmentation field. We first discuss action recognition approaches. Then we overview the efficient action recognition method. Finally, we discuss fully supervised and weakly supervised approaches for temporal action segmentation.

Contents

3.1 Action Recognition

Action recognition and video classification have been the focus of research efforts for an extended period, leading to the development of numerous techniques aimed at generating efficient spatio-temporal feature representations that capture both the appearance and motion propagation within videos. Examples of such techniques include HOG3D (*Klaser et al.*, 2008), SIFT3D (*Scovanner et al.*, 2007), HOF (*Laptev et al.*, 2008), ESURF (*Willems et al.*, 2008), MBH (*Dalal et al.*, 2006), and IDTs (*Wang and Schmid*, 2013). While these approaches have demonstrated promising results, they suffer from the drawback of being manually designed, which can be time-consuming and may not always result in optimal performance. In particular, IDTs, which have achieved the best performance among these techniques, are computationally intensive and lack scalability in terms of capturing semantic concepts. More recently, several other techniques have been proposed (*Fernando et al.*, 2015) to efficiently model temporal structure in video data.

The application of Convolutional Neural Networks (CNNs) in the field of computer vision has resulted in significant progress in a number of challenging tasks, particularly in action recognition, where the use of end-to-end deep CNNs for video classification has outperformed hand-crafted and pre-defined representations by a substantial margin. Examples of such approaches include the two-stream 2D CNN model proposed by *Simonyan and Zisserman* (2014), which utilizes RGB and optical-flow frames to exploit spatial and temporal information, and Temporal Segment Networks (*Wang et al.*, 2016), which achieved notable results by considering longer temporal information from video clips. While the two-stream model introduced by *Simonyan and Zisserman* (2014) was not fully end-to-end trainable, it outperformed the majority of non-deep learning based methods. Other notable 2D CNN-based models for video classification introduced by *Feichtenhofer et al.* (2016); *Karpathy et al.* (2014b); *Lin et al.* (2019); *Diba et al.* (2017).

Despite their computational efficiency and speed, 2D convolutional neural network (CNN) models struggle to infer complex temporal patterns within video clips. This is due to their lack of a temporal or sequential handling module, such as memory blocks or spatial-temporal kernels. To address this issue, 3D CNNs have been introduced (*Tran et al.*, 2015; *Carreira and Zisserman*, 2017; *Wang et al.*, 2018c; *Stroud et al.*, 2018), which can learn both spatial and temporal representation within a single neural network stream. For instance, *Tran et al.* (2015) and *Carreira and Zisserman* (2017) propose 3D versions of the VGG and Inception architectures for large-scale action recognition tasks on the Sports-1M (*Karpathy et al.*, 2014c) and Kinetics (*Kay et al.*, 2017) datasets, respectively. These methods are able to achieve superior performance without relying on pre-extracted motion information, such as Optical-flow, due to the ability of 3D kernels to extract temporal relations between sequential frames. More recently, methods such as SlowFast Networks (*Feichtenhofer et al.*, 2019), DynamoNet (*Diba et al.*, 2019), and X3D (*Feichtenhofer*, 2020) have made significant strides in more efficiently managing spatial-temporal correlations or learning more accurate motion representation for videos.

Recently, the transformer architecture, which was originally introduced in the natural language processing community (*Vaswani et al.*, 2017), has recently demonstrated promising performance on various computer vision tasks (*Dosovitskiy et al.*, 2021; *Touvron et al.*, 2021; *Liu et al.*, 2021; *Zhou et al.*, 2021; *Rao et al.*, 2021b; *Carion et al.*, 2020; *Zheng et al.*, 2021; *Cheng et al.*, 2021; *Yu et al.*, 2021; *Zhao et al.*, 2021). The field of image classification has seen significant progress with the advent of transformers, as demonstrated by ViT (*Dosovitskiy et al.*, 2021), which applies the standard transformer architecture to images by dividing them into a set of non-overlapping patches and generating patch embeddings of lower dimensionality. These embeddings are then augmented with positional embeddings and processed through multiple transformer blocks, with the addition of a learnable class embedding for classification. While ViT has achieved promising results in image classification, it requires a large amount of data for effective generalization. DeiT (*Touvron et al.*, 2021) addresses this limitation through the incorporation of a distillation token that allows the model to learn from a teacher network, resulting in performance superior to that of ViT. LV-ViT (*Jiang et al.*, 2021) proposes a novel objective function for training vision transformers, leading to improved performance. The success of transformers in image classification has led to the development of video vision transformers, such as TimeSformer (*Bertasius et al.*, 2021), which introduces a new architecture for video understanding by extending the self-attention mechanism of standard transformer models to video, with a complexity of $O(T^2S + TS^2)$, where T and S denote temporal and spatial locations, respectively. X-ViT (*Bulat et al.*, 2021) reduces this complexity to $O(TS^2)$ through the proposal of an efficient video transformer. The state-of-the-art approach of *Yan et al.* (2022) achieves top-1 classification accuracy of 89% on the 400 classes of the Kinetics dataset (*Kay et al.*, 2017). The success of transformers has resulted in the development of unified architectures for both video and image classification (*Fan et al.*, 2021a; *Li et al.*, 2022), a notable achievement in the field.

In spite of prior research efforts to enhance the performance and generalizability of video understanding models, deep video models are frequently computationally expensive to both train and use for inference.

3.2 Efficient Action Recognition

There has been a good effort to propose more efficient methods based on 2D and 3D CNNs (*Lin et al.*, 2019; *Lee et al.*, 2018; *Tran et al.*, 2018a; *Xie et al.*, 2018a; *Zolfaghari et al.*, 2018). *Lin et al.* (2019) have introduced a new temporal shift module (TSM) to enhance 2D-ResNet CNN for video classification. TSM has achieved comparable performances with more complex 3D CNN methods but with fewer computations, complexity and made it possible to run action recognition on low-power hardware. In works by *Tran et al.* (2018a) and *Xie et al.* (2018a) authors have proposed different 2D/3D mixture architectures with different setups in ordering the layers. The work of SlowFast (*Feichtenhofer et al.*, 2019) has investigated the resolution trade-offs between the temporal, spatial, and channel axes, and has employed a hybrid approach in order to reduce computation cost. Specifically, SlowFast utilizes a lightweight pathway with high temporal resolution for temporal information modeling, and a more resource-intensive pathway with lower temporal resolution for spatial information modeling. X3D (*Feichtenhofer*, 2020) builds upon this concept by examining the necessity of the lightweight and heavy pathways, and presents a family of efficient video networks as a solution. Several previous works (*Wu et al.*, 2019a,b; *Korbar et al.*, 2019; *Zhu et al.*, 2020) have attempted to reduce the inference time of existing networks through the use of big-little architecture design. In the context of 2D CNNs, *Wu et al.* (2019a) and *Wu et al.* (2019b) employ expensive models to process salient frames, while utilizing lightweight models to process the remaining frames. In contrast, *Korbar et al.* (2019) and *Zhu et al.* (2020) utilize 3D CNNs to process short chunks of frames rather than single frames. *Korbar et al.* (2019) train a secondary, lightweight network to determine which chunks of input frames should be processed by the more expensive 3D CNN. *Zhu et al.* (2020) utilize a fixed scheme in which a subset of input chunks are processed by an expensive 3D CNN, while the remaining chunks are processed by a less resource-intensive 3D CNN, and an RNN is subsequently employed to fuse the outputs of the different 3D CNNs. While these approaches are effective at reducing GFLOPs during inference, they do not reduce the computational cost of the 3D CNNs themselves, and also result in increased training time due to the need to train two networks instead of one.

In addition to efforts to accelerate the inference of convolutional neural networks, research has also been conducted on improving the efficiency of transformer-based models. In the field of natural language processing, Star-Transformer (*Guo et al.*, 2019) reduces the number of connections from the typical n^2 to a more efficient $2n$ by replacing the fully-connected topology with a star-shaped structure. TinyBERT (*Jiao et al.*, 2020) improves efficiency through the distillation of knowledge from a large "teacher" BERT model into a smaller "student" network. PoWER-BERT (*Goyal et al.*, 2020) speeds up the inference process of the BERT model by identifying and eliminating redundant or less-informative tokens based on their estimated importance scores derived from the self-attention weights of the transformer blocks. To reduce the number of FLOPs in character-level language modeling, *Sukhbaatar et al.* (2019) propose a new self-attention mechanism with adaptive attention span. Scaling Transformers (*Jaszczur et al.*, 2021) introduce a novel transformer architecture equipped with sparse variants of standard transformer layers, enabling fast unbatched decoding performance and improved scalability.

To enhance the efficiency of vision transformers, a number of approaches have been proposed. Sparse factorization of the dense attention matrix (*Child et al.*, 2019) reduces its complexity to $O(n\sqrt{n})$ for the task of autoregressive image generation, while *Roy et al.* (2021) sparsify the attention matrix through the use of clustering to only consider the similarities between keys and queries that belong to the same cluster. DynamicViT (*Rao et al.*, 2021a) introduces a prediction module that determines the importance of tokens and discards those that are uninformative for image classification tasks. Hierarchical Visual Transformer (HVT) (*Pan et al.*, 2021b) employs token pooling, similar to the down-sampling of feature maps in convolutional neural networks, to remove redundant tokens. PS-ViT (*Yue et al.*, 2021) utilizes a progressive sampling module to iteratively learn to sample distinctive input tokens, which are then fed into a vision transformer module with fewer encoder layers than ViT. TokenLearner (*Ryoo et al.*, 2021) introduces a learnable tokenization module that can reduce computational cost by learning a small number of important tokens conditioned on the input, and has demonstrated applicability to both image and video understanding tasks. Token Pooling (*Marin et al.*, 2021) down-samples tokens by grouping them into clusters and returning the cluster centers, while a concurrent work (*Liang et al.*, 2022) utilizes a token reorganization method that first identifies the top-k important tokens through computation of token attentiveness, and then fuses less informative tokens. IA-RED^2 (*Pan et al.*, 2021a) presents an interpretability-aware redundancy reduction framework for vision transformers that discards less informative patches in the input data. Many of these approaches improve the efficiency of vision transformers through the incorporation of architectural changes or additional learnable modules that add extra learnable parameters to the networks.

3.3 Temporal Action Segmentation

Temporal action segmentation is a key problem in video understanding that involves the recognition and ordering of multiple actions within untrimmed videos. This is particularly relevant in real-world scenarios, where trimmed video clips are insufficient for understanding the full context of a video. Despite significant progress in the field, the annotation of each frame in a video remains a labor-intensive and expensive task. To address this issue, weakly supervised approaches have been developed for temporal action segmentation that can learn from videos that are only weakly labeled. These methods leverage weak supervision, in the form of the set of actions present in a video, to provide a cost-effective means of annotation. In this review, we first consider fully supervised approaches to temporal action segmentation and then examine methods that utilize weaker levels of supervision.

Fully Supervised Approaches: The task of action segmentation in videos has garnered significant attention in the field, with numerous works addressing the issue (*Spriggs et al.*, 2009; *Kuehne et al.*, 2016b; *Lea et al.*, 2017; *Zhao et al.*, 2017; *Abu Farha and Gall*, 2019). Early approaches for action segmentation utilized techniques such as multi-scale sliding window processing (*Spriggs et al.*, 2009; *Gaidon et al.*, 2013; *Rohrbach et al.*, 2012; *Karaman et al.*, 2014) or Markov models on top of frame-classifiers (*Kuehne et al.*, 2016b; *Lea et al.*, 2016), but these methods often lack strong temporal modeling and have slow inference times. More recent fully-supervised action segmentation approaches have sought to capture long-range temporal dependencies using temporal convolutional networks (TCN) (*Abu Farha and Gall*, 2019; *Lea et al.*, 2017; *Li et al.*, 2020a). There have also been efforts to enhance TCN predictions through the use of graph convolutional networks (*Huang et al.*, 2020), boundary-aware pooling (*Wang et al.*, 2020c; *Ishikawa et al.*, 2021), or hierarchical modeling (*Ahn and Lee*, 2021). In the work by *Gao et al.* (2021), a neural network architecture search approach was employed to determine optimal dilation factors for TCN layers. More recently, the success of transformer-based models in image and video classification motivated *Yi et al.* (2021) to propose a transformer-based architecture for temporal action segmentation.

Weakly Supervised Approaches: The use of action sets as a form of supervision in video labeling has garnered attention in recent years (*Richard et al.*, 2018a; *Li and Todorovic*, 2020, 2021). While *Richard et al.* (2018a) attempted to tackle the problem by assuming the existence of transcripts annotated with all action labels in a video and using alignment to infer frame-wise labeling, this approach is limited in its effectiveness due to the impracticality of aligning all potential transcripts and the fact that it does not utilize the provided annotations directly for learning.

Set supervision provides only the set of actions that occur in the videos without any information regarding the order or how many times each action occurs. However, the performance of these approaches is still inferior compared to transcript supervision. In contrast to set supervision, transcripts provide the ordered list of actions that occur in the video. Transcript based weakly-supervised action segmentation has garnered increasing attention in recent years. *Bojanowski et al.* (2014) introduced the Hollywood extended dataset and proposed a discriminative clustering-based method for action alignment. *Huang et al.* (2016) proposed an extended version of the connectionist temporal classification loss that considers the similarity of frames within the same action. Inspired by techniques used in speech processing, *Kuehne et al.* (2017) introduced a hidden Markov model-based approach and employed a Gaussian mixture model as the observation model. This method iteratively generates pseudo ground truth for videos at the beginning of each epoch, refining it at the end of the epoch. *Richard et al.* (2017); *Ding and Xu* (2018) and *Kuehne et al.* (2020) built upon this work by replacing the Gaussian mixture model with a recurrent neural network for short-range temporal modeling, but retained the iterative nature involving the generation of pseudo ground truth at each epoch. While these methods rely on iterative approaches with two-step optimization that do not permit direct end-to-end training, *Richard et al.* (2018b) introduced the Neural Network Viterbi (NNV) method, which generates pseudo ground truth for each iteration rather than each epoch but requires an additional buffer, thereby increasing training time. NNV also employs a heuristically updated global length model for actions. Recently, *Li et al.* (2019) proposed an extension of NNV that outperformed existing approaches in terms of accuracy on standard benchmarks. They introduced a constrained discriminative loss that discriminates between the energy of valid and invalid segmentations of training videos, resulting in a significant improvement in accuracy compared to NNV.

CHAPTER 4

Spatio-Temporal Channel Correlation for Action Classification

This chapter is based on the following publication:
Spatio-Temporal Channel Correlation Networks for Action Classification
Ali Dina*, Mohsen Fayyaz*, Vivek Sharma, M. Mahdi Arzani, Rahman Yousefzadeh, Juergen Gall, Luc Van Gool
European Conference on Computer Vision (ECCV), 2018.
* *denotes equal contribution.*

Below we will summarize the contributions of each author.

- **Ali Diba and Mohsen Fayyaz**
  The present work represents a collaborative effort between Ali Diba and Mohsen Fayyaz. Ali Diba initially proposed the concept of the STC block and the idea of knowledge transfer, while Mohsen Fayyaz provided the final formulation of the knowledge transfer loss function. Both authors were also instrumental in the implementation and empirical evaluation of the proposed approach. In terms of writing, both Ali Diba and Mohsen Fayyaz contributed significantly to the preparation of the manuscript. It is worth noting that the authors contributed equally to this work.

- **Vivek Sharma, M. Mahdi Arzani, Rahman Yousefzadeh**
  The authors made significant contributions through their ideas, discussions, and writing.

- **Juergen Gall and Luc Van Gool**
  This work was made possible through the supervision and funding provided by both authors. In addition to their financial and managerial support, the authors also contributed through suggestions, discussions, and writing. It is worth noting that both authors made significant contributions to the success of the project.

Contents

In this chapter, we introduce a new block that models correlations between channels of a 3D CNN with respect to temporal and spatial features. This new block can be added as a residual unit to different parts of 3D CNNs. We name our novel block 'Spatio-Temporal Channel Correlation' (STC). By embedding this block to the architectures such as ResNext and ResNet (*Hara et al.*, 2018), we improve the performance on the Kinetics-400Ù HMDB51 (*Kuehne et al.*, 2011), UCF101 (*Soomro et al.*, 2012) and Kinetics (*Kay et al.*, 2017) datasets. The other issue in training 3D CNNs is about training them from scratch with a huge labeled dataset to get reasonable performance. So the knowledge learned in 2D CNNs is completely ignored. Another contribution in this chapter is a simple and effective technique to transfer knowledge from a pre-trained 2D CNN to a randomly initialized 3D CNN for stable weight initialization. This allows us to significantly reduce the number of training samples for 3D CNNs. Thus, by fine-tuning this network, we beat the performance of methods that were trained on large video datasets, *e.g.* Sports-1M (*Karpathy et al.*, 2014c), and fine-tuned on the target datasets, *e.g.* HMDB51/UCF101.

4.1 Introduction

It has been well established that leveraging temporal information in addition to spatial cues can greatly benefit video classification tasks (*Diba et al.*, 2017; *Tran et al.*, 2015; *Yue-Hei Ng et al.*, 2015). In recent years, researchers have focused on improving the modeling of spatio-temporal correlations in 3D CNNs. However, these architectures, like their 2D counterparts, only attempt to learn local correlations within input channels and ignore correlations between channels in both spatial and temporal dimensions, which can hinder their performance. Additionally, training 3D CNNs often requires an abundance of labeled data, which can be computationally expensive and time-consuming. To address these limitations, we propose a new network architecture block that effectively captures both spatial-channel and temporal-channel correlations throughout all layers of the network. We also propose an effective supervision transfer method that enables the transfer of knowledge between different architectures, obviating the need for training the networks from scratch. These contributions significantly improve the computational cost and performance of 3D CNNs for video classification tasks.

In light of the above considerations, we introduce the spatio-temporal channel correlation (STC) block for simultaneous consideration of inter-channel correlations across both spatial and temporal features. The STC block can be incorporated into any set of transformations within a network, such as convolutional layers, for performing spatio-temporal channel correlation feature learning. The STC block consists of two branches: a spatial correlation branch (SCB) and a temporal correlation branch (TCB). The SCB analyzes spatial channel-wise information while the TCB analyzes temporal channel-wise information. The input features $I \in \mathbb{R}^{H\times W\times T\times C}$ are fed into the SCB and TCB, where they undergo global pooling operations to generate representations of the global receptive field. These representations serve two primary purposes: (i) considering global correlations in I by aggregating global features over the input, and (ii) providing a channel-wise descriptor for analyzing inter-channel correlations. The resulting channel-wise feature vectors are then passed through bottleneck fully connected layers to learn dependencies between channels. Output features from both branches are then combined and returned as the output of the STC block, which can be concatenated with the output features of the corresponding layer(s). By leveraging these enriched features in conjunction with traditional features within 3D CNNs, we enhance their representation capabilities. As a result, 3D CNNs equipped with STC blocks are capable of learning channel-wise dependencies and better represent videos. We have applied the STC block to 3D CNN architectures such as 3D-ResNext and 3D-ResNet (*Hara et al.*, 2018), inserting it after each residual block of these networks. Our experiments demonstrate that the use of STC blocks significantly improves the performance of these networks on various video classification tasks.

As previously mentioned, training 3D CNNs from scratch requires a large labeled dataset and can be computationally intensive, taking up to two months to learn a good feature representation on a large-scale dataset such as Sports-1M, which is then fine-tuned on target datasets to improve performance (*Tran et al.*, 2017). To address this issue, we present a method for supervision transfer across architectures, allowing us to avoid the need for training 3D CNNs from scratch altogether. Specifically, we demonstrate that a 2D CNN pre-trained on ImageNet can serve as a "*teacher*" for supervision transfer to a randomly initialized 3D CNN, providing stable weight initialization. Through this transfer learning approach, we outperform the performance of generic 3D CNNs (C3D *Tran et al.* (2015)) trained on Sports-1M and fine-tuned on the HMDB51 and UCF101 datasets. This not only reduces computational workload and training time but also allows us to take advantage of the rich information learned by the 2D CNN on the ImageNet dataset.

4.2 Spatio-Temporal Channel Correlation

The spatio-temporal channel correlation (STC) block is a computational block that can be easily incorporated into any 3D CNN architecture. We demonstrate this by adding the STC block to the ResNet and ResNext 3D CNNs introduced by *Hara et al.* (2018). The STC blocks are inserted after each convolutional block in these architectures to enrich the feature representation. As previously mentioned, the STC block leverages both spatial and temporal information by considering correlations between filters in both dimensions. The input to the STC block consists of feature maps from previous convolution layers.

The STC block has a dual path structure that represents different levels of concepts and information. Each path consists of various modules for embedding and capturing dependencies of channels or filters. Our approach is inspired by the Squeeze-and-Excitation (*Hu et al.*, 2017) method, which uses global average pooling (spatial and temporal) followed by two bottleneck fully connected layers and sigmoid activation function. However, unlike *Hu et al.* (2017), the STC block has two branches, or dual paths: one that analyzes pure channel-wise information and the other that analyzes temporal channel-wise information. Given that we are addressing the task of video classification, it is beneficial to extract meaningful representations in both spatial and temporal dimensions. The STC block captures channel dependency information based on this principle. In the following, we describe both branches and their integration into well-known 3D architectures such as 3D-ResNet (*Hara et al.*, 2018).

Notation. The output feature-maps of the 3D convolutions and pooling kernels at the l^{th} layer extracted for an input video is a tensor $X \in \mathbb{R}^{H \times W \times T \times C}$ where H, W, T and C are the height,

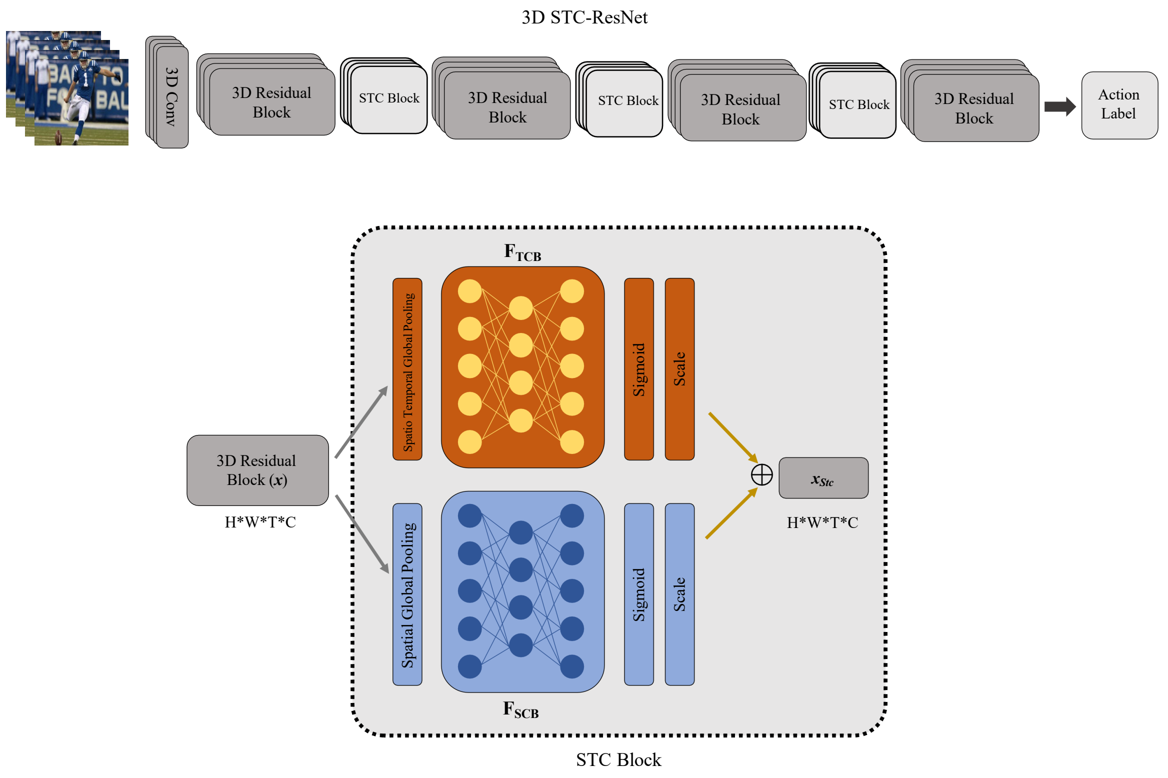


Figure 4.1: **STC-ResNet.** Our STC block is applied to the 3D ResNet. The 3D network uses video clips as input. The 3D feature-maps from the clips are densely propagated throughout the network. The STC operates on the different levels of feature maps in the network to extract spatial and temporal channel relations as new source of information. The output of the network is a video-level prediction.

width, temporal depth and number of channels of the feature maps, respectively. The 3D convolution and pooling kernels are of size ($s \times s \times d$), where d is the temporal depth and s is the spatial size of the kernels.

Temporal Correlation Branch (TCB): In this path the feature map will be squeezed by both spatial and temporal dimensions to extract channel descriptors. If we consider X as the input to STC, the output of the first stage, which is a global spatio-temporal pooling is:

$$z_{tcb} = \frac{1}{W \times H \times T} \sum_{i}^{W} \sum_{j}^{H} \sum_{t}^{T} x_{ijt} \,. \tag{4.1}$$

To obtain the filters non-linear relations, we apply two fully connected layers. The feature dimension is reduced in the first FC layer to C/r (r is reduction ratio) and is increased again to C by the second FC layer. Since we used global spatial-temporal pooling over all dimensions of receptive fields, in the next operation, channel-wise information will be extracted. Right after the sigmoid function, the output of the temporal branch (x_{tcb}) will be calculated by rescaling X using the s_{tcb} vector. So s_{tcb}, output of the bottleneck layers, and x_{tcb}, the branch output, are calculated in this way:

$$s_{tcb} = F_{tcb}(z_{tcb}, W) = W_2(W_1 z_{tcb}) \tag{4.2}$$

$$x_{tcb} = s_{tcb} \cdot X \ . \tag{4.3}$$

W is the parameter set for the bottleneck layers, including $W_1 \in \mathbb{R}^{\frac{C}{r} \times C}$, $W_2 \in \mathbb{R}^{C \times \frac{C}{r}}$ which are FC layers parameters respectively. F_{tcb} is the symbol of fully-connected functions to calculate the s_{tcb}.

Spatial Correlation Branch (SCB): The main difference in this branch compared to the temporal branch is in the aggregation method. The spatial branch shrinks the channel-wise information with respect to the temporal dimension and does global spatial pooling on the input feature map. Therefore this branch is considering the temporal-channel information extraction to enrich the representation in each layer. The calculation of the first operation of the branch comes as following:

$$z_{scb} = \frac{1}{W \times H} \sum_{i}^{W} \sum_{j}^{H} x_{ijT} \tag{4.4}$$

After the pooling layer, we obtain z_{scb} which is a vector with size of $T \times C$. Afterward, there are the fully connected layers to extract the temporal based channel relations. In this branch the first FC layer size is $(T \times C)/r$ and the second FC size is C. Here is the computation description:

$$s_{scb} = F_{scb}(z_{scb}, W) = W_2(W_1 z_{scb}) \tag{4.5}$$

$$x_{scb} = s_{scb} \cdot X \tag{4.6}$$

with $W_1 \in \mathbb{R}^{\frac{(T \times C)}{r} \times (T \times C)}$ and $W_2 \in \mathbb{R}^{C \times \frac{T \times C}{r}}$. By considering both of the branches, the final output of the block (x_{stc}) is computed by averaging over x_{tcb} and x_{scb}.

$$x_{stc} = avg(x_{tcb}, x_{scb}) \tag{4.7}$$

In the case of 3D ResNet or ResNext, this output will be added to the residual layer to have the final output of the Convolution (Conv) blocks.

4.3 Knowledge Transfer

In this section, we propose a method for supervision transfer across architectures, enabling us to circumvent the necessity of training 3D CNNs from scratch. Specifically, we demonstrate that a 2D CNN pre-trained on ImageNet can serve as a "teacher" for supervision transfer to a randomly initialized 3D CNN, providing stable weight initialization. Through this transfer learning approach, we are able to take advantage of the rich information learned by the 2D CNN on the ImageNet dataset, while also avoiding the computational intensity and lengthy training time associated with training 3D CNNs from scratch.

Let I denote a pre-trained 2D CNN that has learned a rich representation from a labeled image dataset, and let V be a randomly initialized 3D CNN using the method of *He et al.* (2015). Our goal is to transfer the knowledge of the representation from I to V in order to provide a stable weight initialization for V. This allows us to avoid the need for training V from scratch, which

Layers	Output Size	3D-ResNet101	3D STC-ResNet101	3D STC-ResNext101
3D Convolution	$56 \times 56 \times 8$	$7 \times 7 \times 7$ conv, stride 2		
3D Pooling	$56 \times 56 \times 8$	$3 \times 3 \times 3$ max pool, stride 1		
Res_1	$28 \times 28 \times 8$	$\begin{bmatrix} conv, 1 \times 1 \times 1, 64 \\ conv, 3 \times 3 \times 3, 64 \\ conv, 1 \times 1 \times 1, 256 \end{bmatrix} \times 3$	$\begin{bmatrix} conv, 1 \times 1 \times 1, 64 \\ conv, 3 \times 3 \times 3, 64 \\ conv, 1 \times 1 \times 1, 256 \\ fc, [16, 256] \end{bmatrix} \times 3$	$\begin{bmatrix} conv, 1 \times 1 \times 1, 128 \\ conv, 3 \times 3 \times 3, 128 \quad C = 32 \\ conv, 1 \times 1 \times 1, 256 \\ fc, [16, 256] \end{bmatrix} \times 3$
Res_2	$14 \times 14 \times 4$	$\begin{bmatrix} conv, 1 \times 1 \times 1, 128 \\ conv, 3 \times 3 \times 3, 128 \\ conv, 1 \times 1 \times 1, 512 \end{bmatrix} \times 4$	$\begin{bmatrix} conv, 1 \times 1 \times 1, 128 \\ conv, 3 \times 3 \times 3, 128 \\ conv, 1 \times 1 \times 1, 512 \\ fc, [32, 512] \end{bmatrix} \times 4$	$\begin{bmatrix} conv, 1 \times 1 \times 1, 256 \\ conv, 3 \times 3 \times 3, 256 \quad C = 32 \\ conv, 1 \times 1 \times 1, 512 \\ fc, [32, 512] \end{bmatrix} \times 4$
Res_3	$7 \times 7 \times 2$	$\begin{bmatrix} conv, 1 \times 1 \times 1, 256 \\ conv, 3 \times 3 \times 3, 256 \\ conv, 1 \times 1 \times 1, 1024 \end{bmatrix} \times 23$	$\begin{bmatrix} conv, 1 \times 1 \times 1, 256 \\ conv, 3 \times 3 \times 3, 256 \\ conv, 1 \times 1 \times 1, 1024 \\ fc, [64, 1024] \end{bmatrix} \times 23$	$\begin{bmatrix} conv, 1 \times 1 \times 1, 512 \\ conv, 3 \times 3 \times 3, 512 \quad C = 32 \\ conv, 1 \times 1 \times 1, 1024 \\ fc, [64, 1024] \end{bmatrix} \times 23$
Res_4	$4 \times 4 \times 1$	$\begin{bmatrix} conv, 1 \times 1 \times 1, 512 \\ conv, 3 \times 3 \times 3, 512 \\ conv, 1 \times 1 \times 1, 2048 \end{bmatrix} \times 3$	$\begin{bmatrix} conv, 1 \times 1 \times 1, 512 \\ conv, 3 \times 3 \times 3, 512 \\ conv, 1 \times 1 \times 1, 2048 \\ fc, [128, 2048] \end{bmatrix} \times 3$	$\begin{bmatrix} conv, 1 \times 1 \times 1, 512 \\ conv, 3 \times 3 \times 3, 512 \quad C = 32 \\ conv, 1 \times 1 \times 1, 2048 \\ fc, [128, 2048] \end{bmatrix} \times 3$
Classification	$1 \times 1 \times 1$	$4 \times 4 \times 1$ avg pool		
Layer		400D softmax		

Table 4.1: 3D ResNet vs. STC-ResNet and STC-ResNext. All the proposed architectures incorporate 3D filters and pooling kernels. Each convolution layer shown in the table corresponds the composite sequence BN-ReLU-Conv operations.

would require a large computational workload and months of training time (*Tran et al.*, 2017). In our proposed method, I serves as a "teacher" for transferring knowledge to the V architecture.

Intuitively, our method uses correspondence between frames and video clips available by the virtue of them appearing together at the same time. Given a pair of X frames and video clip for the same time stamp, the visual information in both frames and video are the same. We leverage this for learning mid-level feature representations by an image-video correspondence task between the 2D and 3D CNN architecture, as depicted in Figure 4.2. We use 2D ResNet (*He et al.*, 2016a) pre-trained on ImageNet (*Deng et al.*, 2009) as I, and the STC-ResNet network as V. The 2D ResNet CNN has 4 convolution blocks and one fully connected layer at the end, while our 3D architecture has 4 3D-convolution blocks with an STC block and we add a fully-connected layer after the last block. We concatenate the last fc layers of both architectures, and connect them with the 2048-dimensional fc layer which is in turn connected to two fully connected layers with 512 and 128 sizes (fc1 , fc2) and to the final binary classifier layer. We use a binary matching classifier: given X frames and a video clip, decide whether the pairs belong to each other or not. For a given pair, X frames are fed sequentially into the network I and we average the last 2D fc features over the X frames, resulting in a 1024-D feature representation. In parallel the video clip is fed to the network V, and we extract the 3D fc features (1024-D), and concatenate them, which is then passed to the fully connected layers for classification. For training, we use a binary classification loss.

During the training, the model parameters of I are frozen, while the task is to effectively learn the model parameters of V without any additional supervision than correspondences between frames and video. The pairs belonging to the same time stamp from the same video are positive pairs, while the pairs coming from two different videos by randomly sampling X frames and video clips from two different videos is a negative pair. Note that, during back-propagation, only the model parameters for V are updated, *i.e.*, transferring the knowledge from I to V. In our experiments we show that a stable

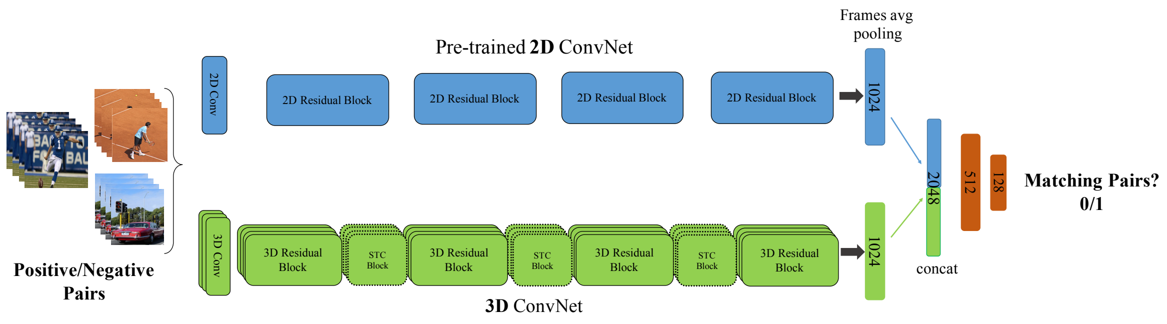


Figure 4.2: **Architecture for knowledge transfer from a pre-trained 2D CNN to a 3D CNN.** The 2D network operates on RGB frames, and the 3D network operates on video clips for the same time stamp. The 2D CNN acts as a teacher for knowledge transfer to the 3D CNN, by teaching the 3D CNN to learn mid-level feature representation by solving an image-video correspondence task. The model parameters of the 2D CNN are frozen, while the task is to effectively learn the model parameters of the 3D CNN only.

weight initialization of V is achieved, and when fine-tuned on the target dataset, it adapts quickly, thus avoiding training the model from scratch. We also show that by using our proposed knowledge transfer method, 3D CNNs can be trained directly on small datasets like UCF101 and achieve a better performance than training from scratch.

Since our transfer learning approach is unsupervised and does not require video labels, we have applied it to a collection of unlabeled videos. Our experiments in Section 9.4 show that our proposed transfer learning approach using STC-ResNext significantly outperforms generic 3D CNNs trained on a large video dataset (*e.g.* Sports-1M) and fine-tuned on target datasets (*e.g.* HMDB51 or UCF101). This demonstrates the effectiveness of our unsupervised transfer learning approach for improving the performance of 3D CNNs on video classification tasks.

4.4 Experiments

In this section, we first introduce the datasets and implementation details of our proposed approach. Afterwards, we provide an extensive study on the architecture of the proposed STC-ResNet and STC-ResNext, which are 3D CNNs. Following, we evaluate and compare our proposed methods with the baselines and other state-of-the-art methods. Finally, we compare our transfer learning: $2D \rightarrow 3D$ CNN performance with generic state-of-the-art 3D CNN methods.

4.4.1 Datasets

We evaluate our proposed method on three challenging video datasets with human actions, namely HMDB51 (*Kuehne et al.*, 2011), UCF101 (*Soomro et al.*, 2012), and Kinetics (*Kay et al.*, 2017). Table 4.2 shows the details of the datasets. For all of these datasets, we use the standard training/testing splits and protocols provided by the datasets. For HMDB51 and UCF101, we report the average accuracy over the three splits, and for Kinetics, we report the performance on the validation and test set.

Kinetics: Kinetics is a new challenging human action recognition dataset introduced by *Kay* *et al.* (2017), which contains 400 action classes. There are two versions of this dataset: untrimmed and trimmed. The untrimmed videos contain the whole video in which the activity is included in a short period of it. However, the trimmed videos contain the activity part only. We evaluate our models on the trimmed version. We use all training videos for training our models from scratch.

UCF101: For evaluating our STC-Nets architectures, we first trained them on the Kinetics dataset, and then fine-tuned them on UCF101. Furthermore, we also evaluate our models by training them from scratch on UCF101 using randomly initialized weights to be able to investigate the effect of pre-training on a huge dataset, such as Kinetics.

HMDB51: Same as UCF101 evaluation we fine-tune the models on HMDB51, which were pre-trained from scratch on Kinetics. Also, we similarly evaluate our models by training them from scratch on HMDB51 using randomly initialized weights.

Data-set	# Clips	# Videos	# Classes
HMDB51 *Kuehne et al.* (2011)	6,766	3,312	51
UCF101 *Soomro et al.* (2012)	13,320	2,500	101
Kinetics *Kay et al.* (2017)	306,245	306,245	400

Table 4.2: Details of the datasets used for evaluation. The 'Clips' shows the total number of short video clips extracted from the 'Videos' available in the dataset.

4.4.2 Implementation Details

We use the PyTorch framework for the implementation and all the networks are trained on 8 Tesla P100 NVIDIA GPUs. Here, we describe the implementation details of our two schemes, 3D CNN architectures and knowledge transfer from 2D to 3D CNNs for stable weight initialization.

STC-Nets.

Training: We train our STC-Nets (STC-ResNet/ResNext) from scratch on Kinetics. Our STC-Net operates on a stack of 16/32/64 RGB frames. We resize the video to 122px when smaller, and then randomly apply 5 crops (and their horizontal flips) of size 112×112. For the network weight initialization, we adopt the same technique proposed in *He et al.* (2015). For the network training, we use SGD, Nesterov momentum of 0.9, weight decay of 10^{-4} and batch size of 128. The initial learning rate is set to 0.1, and reduced by a factor of 10 manually when the validation loss is saturated. The maximum number of epochs for the whole Kinetics dataset is set to 200. Batch normalization also has been applied. The reduction parameter in STC blocks, r, is set to 4.

Testing: For video prediction, we decompose each video into non-overlapping clips of 16/32/64 frames. The STC-Net is applied over the video clips by taking a 112×112 center-crop, and finally we average the predictions over all clips to make a video-level prediction.

Knowledge Transfer: $2D \rightarrow 3D$ **CNNs.** We employ 2D ResNet architecture, pre-trained on ImageNet (*Deng et al.*, 2009), while the 3D CNN is our STC-ResNet network. To the 2D CNN, 16 RGB frames are fed as input. The input RGB images are randomly cropped to the size 112×112, and then mean-subtracted for the network training. To supervise transfer to the STC-ResNet, we replace the previous classification layer of the 2D CNN with a 2-way softmax layer to distinguish between positive and negative pairs. We use stochastic gradient descent (SGD) with mini-batch size of 32 with a fixed weight decay of 10^{-4} and Nesterov momentum of 0.9. For network training, we start with a learning rate set to 0.1 and decrease it by a factor of 10 every 30 epochs. The maximum number of epochs is set to 150. For training data, we use approx. 500K unlabeled videos from YouTube8m dataset (*Abu-El-Haija et al.*, 2016).

4.4.3 Ablation Study on Architecture Design

To evaluate our STC block on 3D CNNs model, we conducted an architecture study and evaluated different configurations. For this work, we mainly focused on 3D versions for ResNet and ResNext with different input size and depth. Our choice is based on the recently presented good performance of these networks in video classification (*Hara et al.*, 2018).

Model Depth: We first analyze the impact of the architecture depth with 3D-ResNet and 3D-ResNext and we have done a series of evaluations on the network size. For the architecture study, the model weights were initialized using the method of *He et al.* (2015).

We employ three different sizes of 3D STC-ResNet; 18, 50, 101 with STC blocks. Evaluations results of these 3D STC-ResNet models are reported in the Table 4.3. As it can be observed, by adding small overhead of STC blocks, STC-Nets can achieve reasonable performance even in a smaller version of ResNet, since our STC-ResNet50 is comparable in accuracy with a regular ResNet101.

Model Depth	Accuracy %
3D-ResNet 101	46.7
STC-ResNet 18	42.8
STC-ResNet 50	46.2
STC-ResNet 101	**47.9**

Table 4.3: Evaluation results of 3D STC-ResNet model with network sizes of 18, 50, and 101 on UCF101 split 1. All models were trained from scratch.

Temporal Input Size: The number of input frames plays a key role in activity recognition. Therefore, we have reported the performance of our 3D STC-ResNet and 3D STC-ResNext with different number of input frames in Table 4.4. Our evaluation shows that longer clips as input will yield better performance, which confirms the observations made in works by *Hara et al.* (2018) and *Carreira and Zisserman* (2017).

TCB vs SCB: We also have studied the impact of the TCB and SCB branches in our STC-Nets. Since each of them considers a different concept in the branch, we evaluated the performance for three settings: SCB only, TCB only, and SCB-TCB combination (STC). In Table 4.5, the importance of the channel correlation branches is shown. As it is shown, incorporating both branches to capture different types of correlations is performing better than SCB or TCB alone.

Model	UCF101	HMDB51
STC-ResNet 101 (16 frames)	90.1	62.6
STC-ResNet 101 (32 frames)	93.2	68.9
STC-ResNet 101 (64 frames)	93.7	70.5
STC-ResNext 101 (16 frames)	92.3	65.4
STC-ResNext 101 (32 frames)	95.8	72.6
STC-ResNext 101 (64 frames)	96.5	74.9

Table 4.4: Evaluation results of STC-ResNet and 3D STC-ResNext models with temporal depths of 16, 32, and 64 frames for all three splits of UCF101 and HMDB51.

Channel Correlation Branch	Accuracy %
SCB	46.1
TCB	47.2
TCB + SCB	**47.9**

Table 4.5: Performance comparison using different channel correlation blocks (TCB vs SCB) for UCF101 split 1.

Frame Sampling Rate: Finding the right configuration of input-frames which are fed to the CNNs for capturing the appearance and temporal information plays a very critical role in temporal CNNs. For this reason, we investigated the impact of the frame sampling rate for the input stream. The STC-ResNet101 has been used for the ablation study on frame sampling rate for training and testing. We evaluate the model by varying the temporal stride of the input frames in the following set {1, 2 ,4, 16}. Table 4.6 presents the accuracy of STC-ResNet101 trained on inputs with different sampling rates. The best results are obtained with a sampling rate of 2, which we also used for other 3D CNNs in the rest of the experiments.

Input Stride	1	2	4	16
Accuracy %	44.6%	**47.9%**	46.8%	40.3%

Table 4.6: Evaluation results of different frame sampling rates for the STC-ResNet101 model. Trained and tested on UCF101 split 1.

4.4.4 Knowledge Transfer

To apply our proposed supervision transfer, we have tested 2D ResNet as basic pre-trained model on ImageNet, while 3D-ResNet and our STC-ResNet are randomly initialized using the method of *He et al.* (2015) and used as target 3D CNNs. We show that a stable weight initialization via transfer learning is possible for 3D CNN architectures, which can be used as a good starting model for training on small datasets like UCF101 or HMDB51. Since the transfer learning pipeline for 3D CNNs have been tested with two different deep architectures (3D-ResNet and STC-Nets), we clearly show the generalization capacity of our method in deep architectures, which can be easily adapted for other deep networks and tasks which use the similar architectures.

3D CNNs	**UCF101**	**HMDB51**
3D-ResNet-Baseline	88.9	61.7
3D-ResNet-Inflation	90.4	62.6
3D-ResNet-**Transfered**	91.3	64.2
STC-ResNet-Baseline	90.1	62.6
STC-ResNet-**Transfered**	92.6	66.1

Table 4.7: Transfer learning results for 3D CNNs by 2D CNNs over all three splits of UCF101 and HMDB51. All models have the same depth of 101.

Table 4.7 shows the results. The baseline is trained from scratch using random initialization. As it is shown, our transfer method performs better than the baseline for the standard 3D-ResNet as well as for our proposed STC-ResNet. Using inflation also improves the baseline, but it is outperformed by our approach. Note that inflation can only be used if the structure of the 2D and 3D network are the same, while our approach allows to transfer the knowledge from any 2D CNN to a 3D CNN, *e.g.*, from 2D-ResNet to the 3D STC-ResNet as in Table 4.7, which is not possible by inflation.

Method	**Top1-Val**	**Top5-Val**
DenseNet3D	59.5	-
Inception3D	58.9	-
C3D* *Hara et al.* (2018)	55.6	-
3D ResNet101 *Hara et al.* (2018)	62.8	83.9
3D ResNext101 *Hara et al.* (2018)	65.1	85.7
RGB-I3D *Carreira and Zisserman* (2017)	68.4	88
STC-ResNet101 (16 frames)	64.1	85.2
STC-ResNext101 (16 frames)	66.2	86.5
STC-ResNext101 (32 frames)	68.7	88.5
SlowFast 8×8 R101+NL *Feichtenhofer et al.* (2019)	78.7	93.5
SlowFast 16×8 R101+NL *Feichtenhofer et al.* (2019)	79.8	93.9
X3D-XXL *Feichtenhofer* (2020)	80.4	94.6
TimeSformer-L *Bertasius et al.* (2021)	80.7	94.7
ViViT-L/16×2 *Bertasius et al.* (2021)	80.6	94.7
MViT-B, 64×3 *Fan et al.* (2021b)	81.2	95.1
X-ViT (16×) *Bulat et al.* (2021)	80.2	94.7
TokenLearner 16at12 (L/16) *Ryoo et al.* (2021)	**82.1**	-

Table 4.8: Comparison results of our models with other state-of-the-art methods on Kinetics-400 dataset. * denotes the pre-trained version of C3D on the Sports-1M. It is important to note that the work discussed in this chapter has been published in ECCV 2018 *Diba et al.* (2018). To provide a comprehensive overview and better visibility of the trends in the field, we have also included recent state-of-the-art methods that have been published subsequent to the publication date of this chapter.

Method	**UCF101**	**HMDB51**
DT+MVSM *Cai et al.* (2014)	83.5	55.9
iDT+FV *Wang and Schmid* (2013)	85.9	57.2
C3D *Tran et al.* (2015)	82.3	56.8
Conv Fusion *Feichtenhofer et al.* (2016)	82.6	56.8
Two Stream *Simonyan and Zisserman* (2014)	88.6	–
TDD+FV *Wang et al.* (2015)	90.3	63.2
RGB+Flow-TSN *Wang et al.* (2016)	94.0	68.5
P3D *Qiu et al.* (2017)	88.6	–
RGB-I3D *Carreira and Zisserman* (2017)	95.6	74.8
RGB+Flow-I3D *Carreira and Zisserman* (2017)	**98.0**	**80.7**
Inception3D	87.2	56.9
3D ResNet 101 (16 frames)	88.9	61.7
3D ResNet 101-Transfered Knowledge	91.3	64.2
3D ResNext 101 (16 frames)	90.7	63.8
STC-ResNext 101 (16 frames)	92.3	65.4
STC-ResNext 101 (64 frames)	96.5	74.9

Table 4.9: Accuracy (%) performance comparison of STC-Nets (STC-ResNet/ResNext) with state-of-the-art methods over all three splits of UCF101 and HMDB51.

4.4.5 Comparison with the state-of-the-art

Finally, after exploring and studying on STC-Net architectures and the configuration of input data and architecture, we compare our STC-ResNet and STC-ResNext with the state-of-the-art methods by pre-training on Kinetics and finetuning on all three splits of the UCF101 and HMDB51 datasets. For UCF101 and HMDB51, we report the average accuracy over all three splits. The results of the supervision transfer technique experiments were reported in the previous part of the experiments.

Table 4.8 shows the result on the Kinetics dataset for STC-Nets compared with state-of-the-art methods. The STC-ResNext101 with 32 frames input depth achieves higher accuracies than RGB-I3D which has an input size of 64 frames.

Table 4.9 shows the results on the UCF101 and HMDB51 datasets for comparison of STC-Nets with other RGB based action recognition methods. Our STC-ResNext101 (64 frames) model outperforms the 3D-ResNet (*Tran et al.*, 2017), Inception3D, RGB-I3D (*Carreira and Zisserman*, 2017) and C3D (*Tran et al.*, 2015) on both UCF101 and HMDB51 and achieves 96.5% and 74.9% accuracy respectively. We also trained Inception3D, a similar architecture to the I3D (*Carreira and Zisserman*, 2017), without using ImageNet on Kinetics and fine-tuned it on UCF101 and HMDB51 to be able to have a fair comparison. As shown in Table 4.9, STC-ResNext performs better than 3D-ResNext by almost 2% on UCF101. Moreover, we note that the state-of-the-art CNNs (*Carreira and Zisserman*, 2017; *Wang et al.*, 2016) use expensive optical-flow maps in addition to RGB input-frames, as in I3D which obtains a performance of 98% on UCF101 and 80% on HMDB51. Because of such high computation needs, we are not able to run similar experiments, but as it can be concluded from Table 4.9, our best RGB model has superior performance than the other RGB based models.

Note that in our work we have not used dense optical-flow maps, and still achieving comparable performance to the state-of-the-art methods *Wang et al.* (2016). This shows the effectiveness of our

STC-Nets to exploit temporal information and spatio-temporal channel correlation in deep CNNs for video clips. This calls for efficient methods like ours instead of computing the expensive optical-flow information (beforehand) which is very computationally demanding, and therefore difficult to obtain for large scale datasets.

4.5 Summary

In this chapter, we introduced a new ‘Spatio-Temporal Channel Correlation’ (STC) block that models correlations between the channels of a 3D CNN. We clearly show the benefit of exploiting spatio-temporal channel correlation features using the STC block. We equipped 3D-ResNet and 3D-ResNext with our STC block and improved the accuracies by 2-3% on the Kinetics dataset. Our STC blocks are added as a residual unit to other parts of networks and learned in an end-to-end manner. The STC feature-maps model the feature interaction in a more expressive and efficient way without an undesired loss of information throughout the network. Our STC-Nets are evaluated on three challenging action recognition datasets, namely HMDB51, UCF101, and Kinetics. The STC-Net architectures achieve state-of-the-art performance on HMDB51, UCF101 and comparable results on Kinetics hen contrasted with prior or concurrent temporal deep neural network models. We expect that the proposed STC blocks will also improve other 3D CNNs. Further, we show the benefit of transfer learning between cross architectures, specifically supervision transfer from 2D to 3D CNNs. This provides a valuable and stable weight initialization for 3D CNNs instead of training them from scratch which is also very expensive. Our transfer learning approach is not limited to transfer supervision between RGB models only, as our approach for transfer learning can be easily adapted for transfer across modalities.

CHAPTER 5

Holistic Video Understanding

This chapter is based on the following publication:
Large Scale Holistic Video Understanding
Ali Dina*, Mohsen Fayyaz*, Vivek Sharma*, Manohar Paluri, Juergen Gall, Rainer Stiefelhagen, Luc Van Gool
European Conference on Computer Vision (ECCV), 2020.
* *denotes equal contribution. The first three authors are listed in alphabetical order.*

Below we will summarize the contributions of each author.

- **Ali Diba, Mohsen Fayyaz, and Vivek Sharma**
  The present study represents a collaborative effort among Ali Diba, Mohsen Fayyaz, and Vivek Sharma. Ali Diba and Mohsen Fayyaz recognized the need for a comprehensive video understanding dataset and subsequently proposed the concept of the dataset, with Ali Diba also proposing the HATNet architecture. Both Ali Diba and Mohsen Fayyaz were involved in the creation of the dataset. Ali Diba, Mohsen Fayyaz, and Vivek Sharma all contributed significantly to the implementation and empirical evaluation of the proposed approach. All authors contributed significantly to the writing of the manuscript and are listed in alphabetical order to reflect their equal contribution to the work.
- **Manohar Paluri and Rainer Stiefelhagen**
  The authors made significant contributions through their ideas, discussions, and writing.
- **Juergen Gall and Luc Van Gool**
  This work was made possible through the supervision and funding provided by both authors. In addition to their financial and managerial support, the authors also contributed through suggestions, discussions, and writing. It is worth noting that both authors made significant contributions to the success of the project.

Contents

In this chapter, we present the Holistic Video Understanding (HVU) dataset, a large-scale dataset for multi-label, multi-task video understanding. While current benchmarks for video recognition have advanced the field significantly, they tend to focus on highly specific tasks such as human action or sports recognition, leaving a gap in the ability to describe the overall content of a video. The HVU dataset aims to address this gap by organizing its annotations hierarchically in a semantic taxonomy, encompassing multiple semantic aspects of dynamic scenes. With approximately 572k videos and 9 million annotations spanning over 3142 labels, the HVU dataset captures a wide range of real-world scenarios, including scenes, objects, actions, events, attributes, and concepts.

To demonstrate the generalization capability of the HVU dataset, we apply it to three challenging tasks: video classification, video captioning, and video clustering. For video classification, we introduce a novel spatio-temporal deep neural network architecture called the Holistic Appearance and Temporal Network (HATNet), which combines 2D and 3D architectures and is trained in an end-to-end manner to address the multi-label, multi-task learning problem. Our experiments validate the idea that holistic representation learning can play a crucial role in enabling many real-world applications.

5.1 Introduction

Video understanding is a complex problem that involves the recognition of various semantic aspects, including the scene or environment, objects, actions, events, attributes, and concepts. Despite significant progress in video recognition, it remains largely limited to action recognition due to the lack of an established video benchmark that integrates the joint recognition of multiple semantic aspects in a dynamic scene. The use CNNs has greatly advanced several subfields of computer vision, however, the training of CNNs for video understanding using a single label per task has been found to be inadequate for describing the content of a video. This limitation hinders the ability of CNNs to learn a generic feature representation for comprehensive video analysis. To address this issue, it is possible to recast the video understanding problem as a multi-task classification task, where multiple labels are assigned to a video from various semantic aspects. This approach is similar to the use of CNNs trained on ImageNet for image classification, which facilitated the learning of a generic feature representation for various vision tasks. Therefore, training CNNs on a dataset comprising

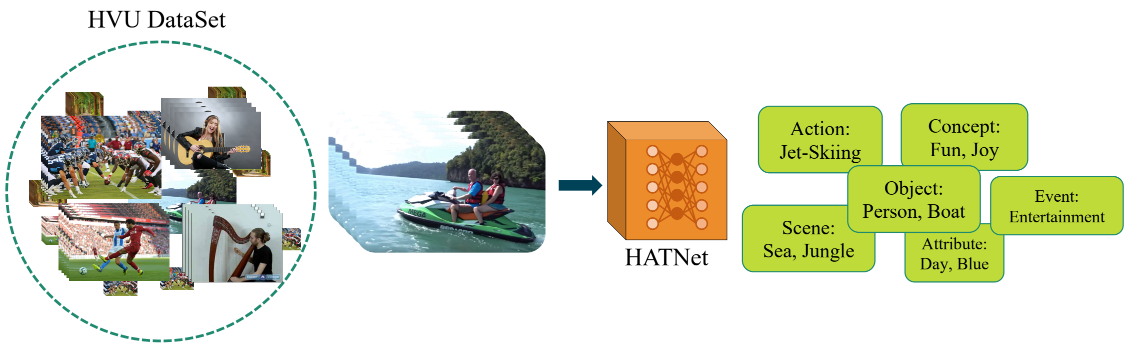


Figure 5.1: Holistic Video Understanding Dataset: A multi-label and multi-task fully annotated dataset and HATNet as a new deep ConvNet for video classification.

multiple semantic aspects can be applied to the holistic recognition and understanding of concepts in video data, enabling a more comprehensive description of video content.

In this work, we introduce the “Holistic Video Understanding Dataset"(**HVU**). HVU is a multi-label, multi-task video benchmark that aims to provide a comprehensive list of tasks and annotations for video analysis and understanding. The dataset is organized hierarchically in a semantic taxonomy and consists of approximately $572k$ videos, with $476k$, $31k$, and $65k$ samples in the train, validation, and test sets, respectively. This makes HVU a sufficiently large dataset, approaching the scale of image datasets. HVU includes approximately $7.5M$ annotations for the training set, $600K$ for the validation set, and $1.3M$ for the test set, spanning over 3142 labels. The recognition of multiple semantic aspects is supported by rich annotations with an average of 2112 annotations per label, including 248 categories for scenes, 1678 for objects, 739 for actions, 69 for events, 117 for attributes, and 291 for concepts. These tasks are based on action recognition datasets (*Gu et al.*, 2018; *Kay et al.*, 2017; *Kuehne et al.*, 2011; *Soomro et al.*, 2012; *Zhao et al.*, 2019) and further extended by incorporating labels for scenes, objects, events, attributes, and concepts in a video. This thorough annotation enables the development of strong algorithms for holistic video understanding that can accurately describe the content of a video. The dataset statistics are summarized in Table 5.1.

To illustrate the significance of holistic representation learning, we demonstrate the impact of HVU on three challenging tasks: video classification, video captioning, and video clustering. To address the task of video classification, we propose a new spatio-temporal architecture called the “Holistic Appearance and Temporal Network" (HATNet). HATNet is designed for multi-label and multi-task learning, enabling the joint solution of multiple spatio-temporal problems simultaneously. HATNet fuses 2D and 3D architectures by combining intermediate representations of appearance and temporal cues, leading to a robust spatio-temporal representation. We evaluate HATNet on the challenging video classification datasets HMDB51, UCF101, and Kinetics, and demonstrate its superior performance. In addition, we show the positive effect of training models using more semantic concepts on transfer learning. Specifically, we demonstrate that pre-training the model on HVU with more semantic concepts improves fine-tuning results on other datasets and tasks compared to pre-training on single semantic category datasets such as Kinetics. This highlights the value of our dataset and the importance of multi-task learning. Furthermore, our experiments on video captioning and video clustering demonstrate the generalization capability of HVU on other tasks, achieving promising results compared to the state-of-the-art. Figure 5.1 shows an overview of our HATNet and

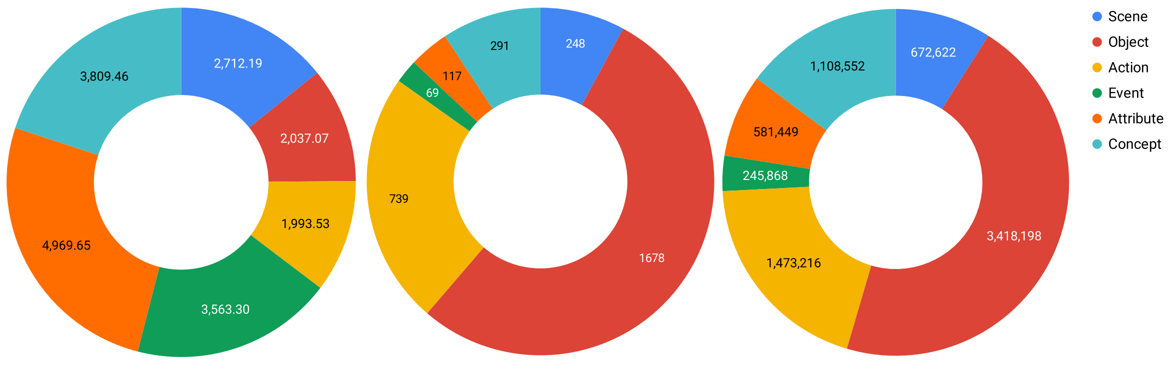


Figure 5.2: Left: Average number of samples per label in each of main categories. Middle: Number of labels for each main category. Right: Number of samples per the main category.

Task Category	Scene	Object	Action	Event	Attribute	Concept	Total
#Labels	248	1678	739	69	117	291	3142
#Annotations	672,622	3,418,198	1,473,216	245,868	581,449	1,108,552	7,499,905
#Videos	251,794	471,068	479,568	164,924	316,040	410,711	481,417

Table 5.1: Statistics of the HVU training set for different categories. The category with the highest number of labels and annotations is the object category.

HVU dataset.

5.2 Holistic Video Understanding Dataset

The HVU dataset is organized hierarchically in a semantic taxonomy of holistic video understanding. While many real-world conditioned video datasets are primarily focused on human action recognition, video content encompasses more than just actions, providing a human-centric description of the video. By focusing solely on human-centric descriptions, we neglect important information about the scene, objects, events, and attributes present in the video. While the SOA dataset (*Ray et al.*, 2018) includes categories for scenes, objects, and actions, to our knowledge it is not publicly available. In comparison, HVU includes a larger number of categories, as shown in Table 5.2. One of the key research questions that has not been adequately addressed in recent works on action recognition is how to leverage other contextual information in a video. The HVU dataset enables the evaluation of the effect of learning and knowledge transfer between tasks, such as enabling transfer learning of object recognition in videos to action recognition and vice versa. In summary, HVU can contribute to the vision community and facilitate more comprehensive solutions for holistic video understanding. Our dataset focuses on the recognition of scenes, objects, actions, attributes, events, and concepts in user-generated videos. The definitions of the scene, object, action, and event categories are consistent with those used in other image and video datasets. The attribute labels describe attributes of scenes, actions, objects, or events. The concept category refers to any noun that presents a grouping definition or higher-level relation in the taxonomy tree for labels of other categories.

Dataset	Scene	Object	Action	Event	Attribute	Concept	#Videos	Year
HMDB51 *Kuehne et al.* (2011)	-	-	51	-	-	-	7K	’11
UCF101 *Soomro et al.* (2012)	-	-	101	-	-	-	13K	’12
ActivityNet *Caba Heilbron et al.* (2015)	-	-	200	-	-	-	20K	’15
AVA *Gu et al.* (2018)	-	-	80	-	-	-	57.6K	’18
Something-Something *Goyal et al.* (2017b)	-	-	174	-	-	-	108K	’17
HACS *Zhao et al.* (2019)	-	-	200	-	-	-	140K	’19
Kinetics *Smaira et al.* (2020)	-	-	700	-	-	-	650K	’20
Moments in Time *Monfort et al.* (2019)	-	-	339	-	-	-	835K	’19
AViD *Piergiovanni and Ryoo* (2020)	-	-	887	-	-	-	450K	’20
EPIC-KITCHEN *Damen et al.* (2018)	-	323	149	-	-	-	39.6K	’18
SOA *Ray et al.* (2018)	49	356	148	-	-	-	562K	’18
HVU (**Ours**)	248	1678	739	69	117	291	572K	’20

Table 5.2: Comparison of the HVU dataset with other publicly available video recognition datasets in terms of #labels per category. Note that SOA is not publicly available.

5.2.1 HVU Statistics

HVU is comprised of $572k$ videos, with $481k$, $31k$, and $65k$ video clips in the train, validation, and test sets, respectively. The dataset includes trimmed video clips with durations ranging up to 10 seconds. HVU has 6 main categories: scene, object, action, event, attribute, and concept, totaling 3142 labels with approximately $7.5M$ annotations for the train, validation, and test sets. On average, there are approximately 2112 annotations per label. The distribution of categories in terms of annotations, labels, and annotations per label is depicted in Figure 5.2. It can be observed that the object category has the highest number of labels and annotations, which is due to the abundance of objects in video. While the object category has the highest number of labels and annotations, it does not have the highest ratio of annotations per label. The average of 2112 annotations per label is a reasonable amount of training data for each label. The scene and action categories have relatively lower numbers of labels and annotations due to the trimmed videos and short durations of the videos in the dataset. The dataset statistics for each category in the training set are shown in Table 5.1.

5.2.2 Collection and Annotation

Constructing a large-scale video understanding dataset is a labor-intensive process, primarily due to the two main tasks involved: data collection and data annotation. Many popular datasets, such as ActivityNet, Kinetics, and YouTube-8M, are collected from sources like YouTube and are annotated using a semi-automatic crowdsourcing strategy in which a human manually verifies the videos. We adopted a similar approach with technical differences in order to reduce the cost of data collection and annotation. To ensure diversity in the taxonomy of HVU, we used YouTube-8M (*Abu-El-Haija et al.*, 2016), Kinetics-600 (*Carreira et al.*, 2018), and HACS (*Zhao et al.*, 2019) as the main sources for our dataset. This also allowed us to avoid copyright and privacy issues, enabling us to publicly release the dataset. Additionally, using these datasets as sources ensured that none of the test videos from these datasets were included in the training set of HVU. It is worth noting that all of the aforementioned datasets are action recognition datasets.

Manually annotating a large number of videos with multiple semantic categories (*e.g.*, thousands of concepts and tags) is prone to error and is a time-consuming task due to the volume and duration of the videos. To address these issues, we developed a two-stage framework for HVU annotation. In the first stage, we used the Google Vision API (*goo*) and Sensifai Video Tagging API (*sen*) to obtain rough annotations of the videos. These APIs predicted 30 tags per video, with the probability threshold set relatively low (approximately 30%) to avoid false rejections of tags. The tags were chosen from a dictionary of approximately $8k$ words. This process resulted in almost 18 million tags for the entire dataset. In the second stage, we performed human verification to remove any potentially mislabeled noisy tags and also added any missing tags that were missed by the APIs, based on the recommended tags of similar videos. This human annotation step resulted in 9 million tags for the entire dataset, with approximately 3500 different tags.

5.2.3 Human Annotation Details

The present study utilized human annotators to validate machine-generated annotations. The initial set of machine-generated annotations comprised approximately $8k$ labels, which were subsequently refined through multiple stages of human verification. In the first stage, 4378 labels were retained, and in the final stage of the annotation process, an additional 80 labels were added by human annotators, resulting in a total of 3142 labels. It is worth noting that the human annotation process not only served to verify the accuracy of the machine-generated labels but also afforded the opportunity to enrich the dataset through the introduction of novel labels.

In specific for the HVU human verification task, we employed three different teams (Team-A, Team-B and Team-C) of 55 human annotators. Team-A focused on constructing a taxonomy for the dataset based on the visual meaning and definitions of the tags provided by APIs. Team-B and Team-C were responsible for verifying the tags and videos through four tasks: (a) flagging false tags by watching each video, (b) reviewing the tags by watching the videos for each tag and flagging any incorrect videos, (c) adding missing tags to the videos, and (d) suggesting modifications to the tags, such as renaming or merging.

To ensure that Team-B and Team-C had a thorough understanding of the tags and corresponding videos, they were provided with the tag definitions from Team-A. Team-B first performed the aforementioned tasks on all videos and provided the initial round of clean annotations, which were subsequently reviewed by Team-C to ensure accuracy. Finally, suggestions from tasks (c) and (d) were reviewed by Team-A and applied to the dataset as necessary. The verification process takes $\sim$100 seconds on average per video clip for a trained worker. It took about 8500 person-hours to firstly clean the machine-generated tags and remove errors and secondly add any possible missing labels from the dictionary. By incorporating the machine generated tags and human annotation, the HVU dataset covers a diverse set of tags with clean annotations. Using machine generated tags in the first step helps us to cover a larger number of tags than a human can remember and label them in a reasonable time. To maintain a balanced distribution of samples per tag, a minimum of 50 samples per tag were required.

In order to provide a more comprehensive overview of the HVU human annotation process, we present the statistics of the various stages of the annotation process in Table 5.3. It is worth noting that the labels and categories listed in the table are the results of the initial human annotation process on the validation set of the dataset. As can be seen in the table, the category with the highest number of labels and annotations is the object category, while the concept category has the lowest number of labels.

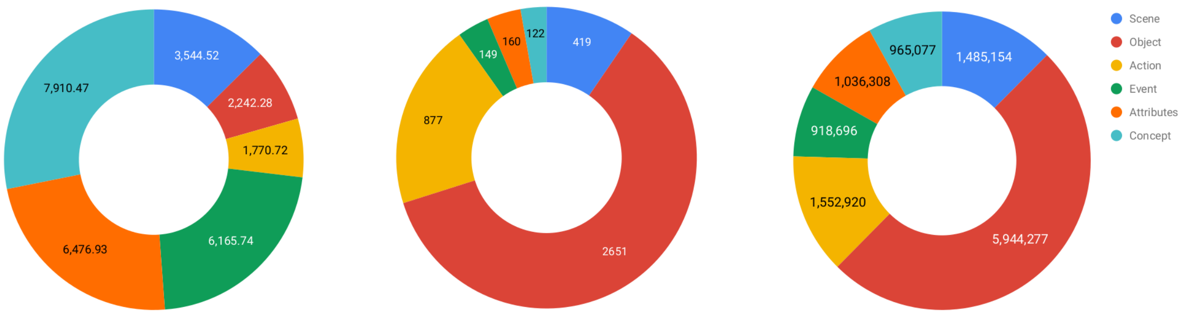


Figure 5.3: Left: Average number of samples per label in each of main categories. Middle: Number of labels for each main category. Right: Number of samples per the main category. All statistics are for the machine generated tags of the HVU training set.

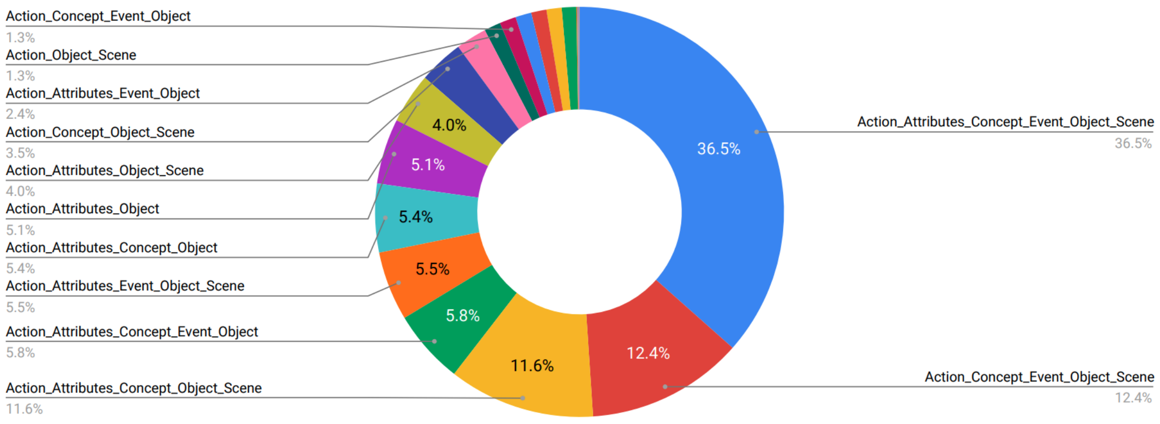


Figure 5.4: Coverage of different subsets of the 6 main semantic categories in videos. 16.4% of the videos have annotations of all categories. All statistics are for the machine generated tags of the HVU training set.

To further illustrate the distribution of categories, we depict the number of annotations, labels, and annotations per label for each category in Figure 5.3. As expected, the object category exhibits the highest number of labels and annotations due to the abundance of objects in the videos. However, it is worth noting that the object category does not have the highest ratio of annotations per label.

In Figure 5.4, we present the percentage of videos belonging to each subset of the main categories. In total, there are 50 different subsets of videos based on the assigned semantic categories. Approximately 36% of the videos belong to all of the categories. We present some samples of videos and their corresponding tags in Fig 5.7 and Fig 5.8.

5.2.4 Taxonomy

As previously mentioned, the Google and Sensifai APIs were utilized to predict tags for the videos in the dataset, resulting in a preliminary set of approximately 8K tags. These tags covered a wide range of categories, including scenes, objects, events, attributes, concepts, logos, emotions, and actions. In order to clean and refine this set of tags, we employed the WordNet ontology (*Miller*, 1995) and

Task Category	Scene	Object	Action	Event	Attribute	Concept	Total
#Labels	419	2651	877	149	160	122	4378
#Annotations	1,485,154	5,944,277	1,552,920	918,696	1,036,308	965,077	11,902,432
#Videos	366,941	480,821	481,418	320,428	368,668	375,664	481,418

Table 5.3: Statistics of machine generated tags of HVU training set for different categories. The category with the highest number of labels and annotations is the object category.

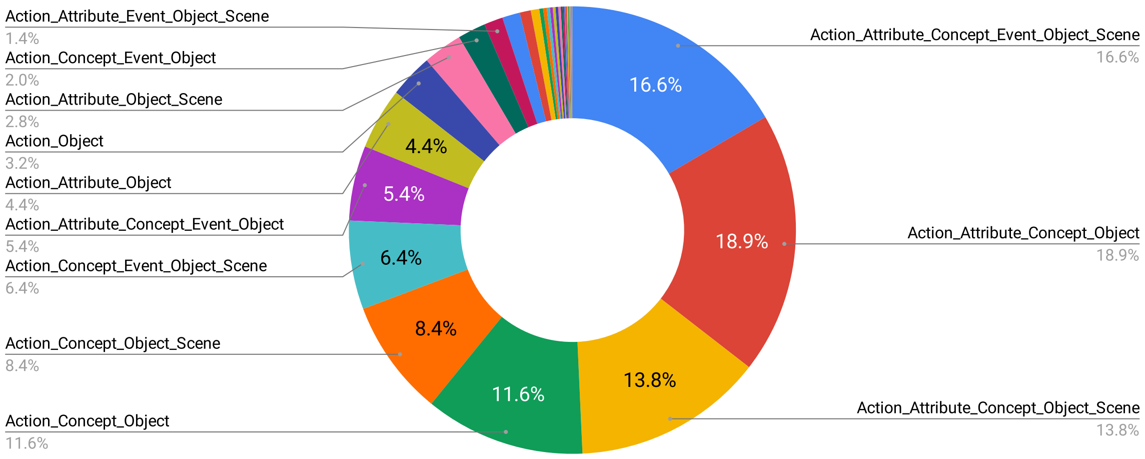


Figure 5.5: Coverage of different subsets of the 6 main semantic categories in videos. 16.6% of the videos have annotations of all categories.

removed tags with imbalanced distribution. The refinement and pruning process aimed to preserve the true distribution of labels and resulted in the final taxonomy. This taxonomy was subsequently classified into six main semantic categories (scenes, objects, actions, events, attributes, and concepts) by human annotators.

It is worth noting that each video may be assigned to multiple semantic categories. In fact, nearly 100K of the videos in the HVU dataset belong to all of the semantic categories. In comparison to SOA, almost half of HVU videos have labels for scene, object, and action together. The distribution of videos among the various subsets of the main categories is depicted in Figure 5.5.

5.3 Holistic Appearance And Temporal Network

In this section, we first provide a brief overview of state-of-the-art 3D CNNs for video classification. We then introduce our proposed "Holistic Appearance and Temporal Network" (HATNet) for multi-task and multi-label video classification.

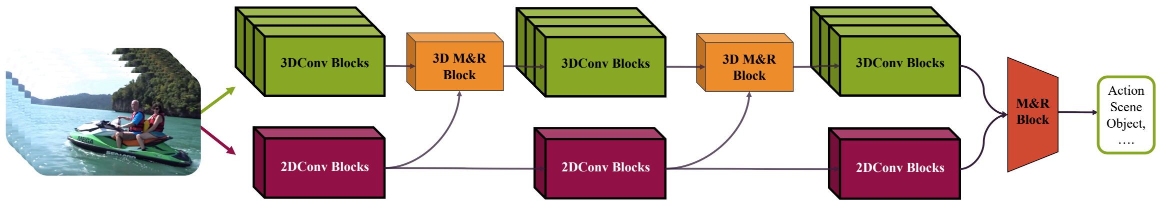


Figure 5.6: HATNet: A new 2D/3D deep neural network with 2DConv, 3DConv blocks and merge and reduction (M&R) block to fuse 2D and 3D feature maps in intermediate stages of the network. HATNet combines the appearance and temporal cues with the overall goal to compress them into a more compact representation.

5.3.1 3D-CNNs Baselines

3D CNNs are designed to capture temporal cues in video and have demonstrated efficient performance for video classification tasks. These networks simultaneously process spatial and temporal information. In this study, we utilize 3D-ResNet (*Tran et al.*, 2017) and STCnet (chapter 4) as our 3D CNN baselines, which have achieved competitive results on the Kinetics and UCF101 datasets. To evaluate the performance of these methods on the multi-label HVU dataset, we employ mean average precision (mAP) across all labels and report individual performance for each category. The results of these methods are presented in Table 5.5 and were obtained using binary cross entropy loss.

5.3.2 Multi-Task Learning 3D-CNNs

Another approach that is studied in this work to tackle the HVU dataset is to have the problem solved with multi-task learning or a joint training method. As we know the HVU dataset consists of high-level categories like objects, scenes, events, attributes, and concepts, so each of these categories can be dealt like separate tasks. In our experiments, we have defined six tasks, scene, object, action, event, attribute, and concept classification. So our multi-task learning network is trained with six objective functions, that is with multi-label classification for each task. The trained network is a 3D-CNN which has separate Conv layers as separate heads for each of the tasks at the end of the network. For each head we use the binary cross entropy loss since it is a multi-label classification for each of the categories.

5.3.3 2D/3D HATNet

Our proposed HATNet is a novel spatio-temporal neural network designed to efficiently handle multi-task and multi-label video classification by maximally engaging both temporal and appearance information. The motivation for HATNet arises from the need for a deep neural network capable of recognizing different levels of concepts in holistic video recognition, including still objects, dynamic scenes, various attributes, and human activities. HATNet employs a flexible approach, utilizing both a 2D pre-trained model trained on a large image dataset such as ImageNet and a 3D pre-trained model trained on video datasets such as Kinetics to accelerate the training process. However, HATNet can also be trained from scratch, as demonstrated in our experiments. HATNet possesses the ability to learn a hierarchical spatio-temporal feature representation through the use of appearance and temporal neural modules.

Appearance Neural Module. In the design of HATNet, we employ 2D CNNs with 2D convolutional (2DConv) blocks to extract static visual cues from individual frames in a video clip. The inclusion of this module in the network is crucial for our goal of recognizing objects, scenes, and attributes alongside actions within the video. Specifically, the 2DConv blocks are utilized to capture the spatial structure present in the frame, enabling the network to handle the aforementioned concepts more effectively.

Temporal Neural Module. In HATNet architecture, the 3D Convolutions (3DConv) module handles temporal cues dealing with interaction in a batch of frames. 3DConv aims to capture the relative temporal information between frames. It is crucial to have 3D convolutions in the network to learn relational motion cues for efficiently understanding dynamic scenes and human activities. We use ResNet18/50 for both the 3D and 2D modules, so that they have the same spatial kernel sizes, and thus we can combine the output of the appearance and temporal branches at any intermediate stage of the network.

Figure 5.6 shows how we combine the 2DConv and 3DConv branches and use merge and reduction blocks to fuse feature maps at the intermediate stages of HATNet. Intuitively, combining the appearance and temporal features are complementary for video understanding and this fusion step aims to compress them into a more compact and robust representation. In the experiment section, we discuss in more detail the HATNet design and how we apply merge and reduction modules between 2D and 3D neural modules. Supported by our extensive experiments, we show that HATNet complements holistic video recognition, including understanding the dynamic and static aspects of a scene and also human action recognition. In our experiments, we have also performed tests on HATNet based multi-task learning similar to 3D-CNNs based multi-task learning discussed in Section (5.3.2). HATNet has some similarities to the SlowFast (*Feichtenhofer et al.*, 2019) network but there are major differences. SlowFast uses two 3D-CNN networks for a slow and a fast branch. HATNet has one 3D-CNN branch to handle motion and dynamic information and one 2D-CNN to handle static information and appearance. HATNet also has skip connections with M&R blocks between 3D and 2D convolutional blocks to exploit more information.

2D/3D HATNet Design. The HATNet architecture includes two branches: a 3D convolutional (3DConv) branch and a 2D convolutional (2DConv) branch. Following each block in the 3DConv and 2DConv branches, a merging and reduction block is applied. This involves concatenating the feature maps produced by the block and performing channel reduction via a $1 \times 1 \times 1$ convolution. For example, given the feature maps produced by the first block in both the 3DConv and 2DConv branches, each with 64 channels, the maps are first concatenated to result in 128 channels. A $1 \times 1 \times 1$ convolution is then applied using 64 kernels, resulting in an output with 64 channels. This merging and reduction process is independently performed within both the 3DConv and 2DConv branches, continuing until the final merging of the two branches.

We employ 3D-ResNet and STCnet (chapter 4) with ResNet18/50 as the HATNet backbone in our experiments. The STCnet is a model of 3D networks with spatio-temporal channel correlation modules which improves 3D networks performance significantly. We also had to make a small change to the 2D branch and remove pooling layers right after the first 2D Conv to maintain a similar feature map size between the 2D and 3D branches since we use 112×112 as input-size.

Dataset	Scene	Object	Action	Event	Attribute	Concept	HVU Overall %
Machine-Generated HVU	46.3	22.4	43.8	31.4	25.3	20.1	31.6
Human-Annotation HVU	50.1	27.9	46.7	35.7	29.2	23.2	35.4

Table 5.4: mAP (%) Performance comparison between machine generated and human-verified tags of HVU. This evaluation shows how the human annotation process is crucial to have a more efficient dataset. The CNN model which is used for this experiment is 3D-ResNet18.

Model	Scene	Object	Action	Event	Attribute	Concept	HVU Overall %
3D-ResNet	50.6	28.6	48.2	35.9	29	22.5	35.8
3D-STCNet	51.9	30.1	50.3	35.8	29.9	22.7	36.7
HATNet	**55.8**	**34.2**	**51.8**	**38.5**	**33.6**	**26.1**	**40**

Table 5.5: mAP (%) performance of different architecture on the HVU dataset. The backbone CNN for all models is ResNet18.

5.4 Experiments

In this section, we demonstrate the effectiveness of HVU on three tasks: video classification, video captioning, and video clustering. First, we present the implementation details and evaluate the performance of HVU on multi-label video recognition. We then compare the transfer learning ability of HVU to that of Kinetics. As an additional experiment, we demonstrate the importance of incorporating categories such as scenes and objects for video classification. Finally, we examine the generalization capability of HVU on the video captioning and clustering tasks. For each task, we compare our method to state-of-the-art techniques on benchmark datasets. In all experiments, we utilize RGB frames as input to the CNN and employ PyTorch for implementation. For training, we use 16 or 32 frame-long video clips as a single input and train the networks on a machine equipped with 8 V100 NVIDIA GPUs.

5.4.1 Effect of Human Annotation

To present the impact of the human annotation process, we have evaluated both versions of the HVU with machine-generated tags and human-annotated tags. We have trained two 3D-ResNet18 for each set and the comparison came in Table 5.4.

5.4.2 HVU Results

Table 5.5 presents the overall performance of various simpler or multi-task learning baselines and HATNet on the HVU validation set, measured in terms of mean average precision on all labels/tags. HATNet, which jointly considers both appearance and temporal information, achieves the best performance due to its incorporation of an appearance module not present in the other baselines. The success of HATNet demonstrates the utility of combining 3D (temporal) and 2D (appearance) convolutional blocks for learning a more robust reasoning ability.

Model	Scene	Object	Action	Event	Attribute	Concept	Overall
3D-ResNet (Standard)	50.6	28.6	48.2	35.9	29	22.5	35.8
HATNet (Standard)	55.8	34.2	51.8	38.5	33.6	26.1	40
3D-ResNet (Multi-Task)	51.7	29.6	48.9	36.6	31.1	24.1	37
HATNet (Multi-Task)	**57.2**	**35.1**	**53.5**	**39.8**	**34.9**	**27.3**	**41.3**

Table 5.6: Multi-task learning performance (mAP (%) comparison of 3D-ResNet18 and HATNet, when trained on HVU with all categories in the multi-task pipeline. The backbone CNN for all models is ResNet18.

Pre-Training Dataset	UCF101	HMDB51	Kinetics
From Scratch	65.2	33.4	65.6
Kinetics	89.8	62.1	-
HVU	**90.5**	**65.1**	**67.8**

Table 5.7: Performance (mAP (%)) comparison of HVU and Kinetics datasets for transfer learning generalization ability when evaluated on different action recognition datasets. The trained model for all of the datasets is 3D-ResNet18.

5.4.3 Multi-Task Learning on HVU

Since the HVU is a multi-task classification dataset, it is interesting to compare the performance of different deep neural networks in the multi-task learning paradigm as well. For this, we have used the same architecture as in the previous experiment, but with a different last layer of convolutions to observe multi-task learning performance. We have targeted six tasks: scene, object, action, event, attribute, and concept classification. In Table 5.6, we have compared standard training without multi-task learning heads versus multi-task learning networks.

As expected, the simple baseline multi-task learning methods achieve higher performance on individual tasks as expected, in comparison to standard networks learning for all categories as a single task. Therefore this initial result on a real-world multi-task video dataset motivates the investigation of more multi-task learning methods for video classification.

5.4.4 Transfer Learning: HVU vs Kinetics

In this experiment, we study the ability of transfer learning with the HVU dataset. We compare the results of pre-training 3D-ResNet18 using Kinetics versus using HVU and then fine-tuning on UCF101, HMDB51, and Kinetics. Obviously, there is a large benefit from pre-training of deep 3D-CNNs and then fine-tune them on smaller datasets (i.e. HVU, Kinetics $\Rightarrow$ UCF101 and HMDB51). As it can be observed in Table 5.7, models pre-trained on our HVU dataset performed notably better than models pre-trained on the Kinetics dataset. Moreover, pre-training on HVU can improve the results on Kinetics also.

Training Labels	Action Recognition Performance
Action Only	65.6
Action, Scene, Object, Event, Attribute, Concept	68.8

Table 5.8: Evaluation of training a 3D-ResNet18 model on Kinetics with HVU labels.

5.4.5 Benefit of Multiple Semantic Categories

In this experiment, we investigate the impact of training models with multiple semantic categories, as opposed to using only a single category such as Kinetics, which focuses exclusively on action. To this end, we design an experiment in which the model is trained in multiple steps, with different categories of tags added one by one. Specifically, we first train 3D-ResNet18 with action tags from HVU, followed by object tags in the second step and scene tags in the final step. To evaluate performance, we consider the action category of HVU. In the first step, the achieved accuracy was 43.6%, which increased to 44.5% in the second step and 45.6% in the final step. These results demonstrate that the inclusion of high-level categories in the training process boosts the performance for action recognition at each step. As shown in Table 5.6, training all categories together yields 47.5% accuracy for the action category, a gain of approximately 4% over training with only the action category. This suggests that incorporating additional categories allows for the learning of an effective feature representation and facilitates a more comprehensive understanding of the video.

To further understand the effect of multiple semantic categories on action recognition performance, we compare the performance of a 3D-ResNet18 trained on Kinetics-600 videos using two different annotation setups. In the first setup, we use the human action recognition labels of the Kinetics-600 dataset. Since Kinetics-600 is a subset of HVU, in the second setup we utilize the HVU annotations, which include five additional categories in addition to the human action category. We then evaluate the models on Kinetics human action recognition labels. As shown in Table 5.8, incorporating more semantic labels in the training for Kinetics leads to improved action classification performance. This can be attributed to the fact that HVU provides additional capabilities for deep models to learn new visual features and gain a deeper understanding of videos.

5.4.6 Comparison on UCF, HMDB, Kinetics

In Table 5.9, we compare the HATNet performance with the state-of-the-art on UCF101, HMDB51, and Kinetics. For our baselines and HATNet, we employ pre-training in two separate setups: one with HVU and another with Kinetics, and then fine-tune on the target datasets. For UCF101 and HMDB51, we report the average accuracy over all three splits. We have used ResNet18/50 as the backbone model for all of our networks with 16 and 32 input frames. HATNet pre-trained on HVU with 32 frames input achieved superior performance on all three datasets with standard network backbones. Note that on Kinetics, HATNet even with ResNet18 as a backbone CNN performs almost comparable to SlowFast which is trained by dual 3D-ResNet50. In Table 5.9, however, while SlowFast has better performance using dual 3D-ResNet101 architecture, HATNet obtains comparable results with a much smaller backbone.

Method	Pre-Trained Dataset	Backbone	UCF101	HMDB51	Kinetics-400	Kinetics-600
Two Stream (spatial stream) *Simonyan and Zisserman* (2014)	Imagenet	VGG-M	73	40.5	-	
RGB-I3D *Carreira and Zisserman* (2017)	Imagenet	Inception v1	84.5	49.8	-	
C3D *Tran et al.* (2015)	Sport1M	VGG11	82.3	51.6	-	
TSN *Wang et al.* (2016)	Imagenet,Kinetics	Inception v3	93.2	-	72.5	
RGB-I3D *Carreira and Zisserman* (2017)	Imagenet,Kinetics	Inception v1	95.6	74.8	72.1	
3D ResNext 101 (16 frames) *Hara et al.* (2018)	Kinetics	ResNext101	90.7	63.8	65.1	
STC-ResNext 101 (64 frames) (chapter 4)	Kinetics	ResNext101	96.5	74.9	68.7	
ARTNet *Wang et al.* (2018b)	Kinetics	ResNet18	93.5	67.6	69.2	
R(2+1)D *Tran et al.* (2018b)	Kinetics	ResNet50	96.8	74.5	72	
ir-CSN-101 *Tran et al.* (2019)	Kinetics	ResNet101	-	-	76.7	
DynamoNet *Diba et al.* (2019)	Kinetics	ResNet101	-	-	76.8	
SlowFast 4×16 *Feichtenhofer et al.* (2019)	Kinetics	ResNet50	-	-	75.6	78.8
SlowFast 16×8* *Feichtenhofer et al.* (2019)	Kinetics	ResNet101	-	-	78.9*	81.1
HATNet (32 frames)	Kinetics	ResNet50	96.8	74.8	77.2	80.2
X3D-XL *Feichtenhofer* (2020)	Kinetics	X3D	-	-	79.1	81.9
TimeSformer-L *Bertasius et al.* (2021)	ImageNet-21K	ViT	-	-	80.7	82.2
ViViT-L/16×2 *Arnab et al.* (2021)	Kinetics	ViT	-	-	80.6	82.9
MViT-B, 32×3 *Fan et al.* (2021b)	Kinetics	ViT	-	-	80.2	83.4
X-ViT (16×) *Bulat et al.* (2021)	Kinetics	ViT	-	-	80.2	84.5
TokenLearner 16at12 (L/16) *Ryoo et al.* (2021)	Kinetics	ViT	-	-	82.1	84.4
HATNet (32 frames)	HVU	ResNet18	96.9	74.5	74.2	77.4
HATNet (16 frames)	HVU	ResNet50	96.5	73.4	76.3	79.4
HATNet (32 frames)	HVU	ResNet50	**97.8**	**76.5**	**79.3**	**81.6**

Table 5.9: State-of-the-art performance comparison on UCF101, HMDB51 test sets and Kinetics validation set. The results on UCF101 and HMDB51 are average mAP over three splits, and for Kinetics(400,600) is Top-1 mAP on the validation set. For a fair comparison, we report the performance of methods that utilize only RGB frames as input. It is important to note that the work discussed in this chapter has been published in ECCV 2020 *Diba et al.* (2020). To provide a comprehensive overview and better visibility of the trends in the field, we have also included recent state-of-the-art methods that have been published subsequent to the publication date of this chapter. *SlowFast uses multiple branches of 3D-ResNet with bigger backbones.

5.4.7 Video Captioning

We present a second task that showcases the effectiveness of our HVU dataset, we evaluate the effectiveness of HVU for the video captioning task. We conduct experiments on a large-scale video captioning dataset, namely MSR-VTT (*Xu et al.*, 2016). We follow the standard training/testing splits and protocols provided originally in the work by *Xu et al.* (2016). For video captioning, the performance is measured using the BLEU metric.

Method and Results: Most of the state-of-the-art video captioning methods use models pre-trained on Kinetics or other video recognition datasets. With this experiment, we intend to show another generalization capability of the HVU dataset where we evaluate the performance of pre-trained models trained on HVU against Kinetics. For our experiment, we use the Controllable Gated Network (*Wang et al.*, 2019) method, which is to the best of our knowledge the state-of-the-art for captioning task.

For comparison, we considered two models of 3D-ResNet50, pre-trained on (i) Kinetics and (ii) HVU. Table 5.10 shows that the model trained on HVU obtained better gains in comparison to Kinetics. This shows HVU helps to learn more generic video representation to achieve better performance in other tasks.

Model	Pre-Training Dataset	BLEU@4
SA(VGG+C3D) *Xu et al.* (2016)	ImageNet+Sports1M	36.6
M3(VGG+C3D) *Wang et al.* (2018a)	ImageNet+Sports1M	38.1
SibNet(GoogleNet) *Liu et al.* (2018)	ImageNet	40.9
MGSA(Inception+C3D) *Chen and Jiang* (2019)	ImageNet+Sports1M	42.4
I3D+M *Wang et al.* (2019)	Kinetics	41.7
3D-ResNet50+M	Kinetics	41.8
3D-ResNet50+M	HVU	**42.7**

Table 5.10: Captioning performance comparisons of the method by *Wang et al.* (2019) with different models and pre-training datasets. M denotes the motion features from optical flow extracted as in the original paper.

Model	Pre-Training Dataset	Clustering Accuracy (%)
3D-ResNet50	Kinetics	50.3
3D-ResNet50	HVU	53.5
HATNet	HVU	**54.8**

Table 5.11: Video clustering performance: an evaluation based on extracted features from networks pre-trained on Kinetics and HVU datasets.

5.4.8 Video Clustering

With this experiment, we evaluate the effectiveness of generic features learned using HVU when compared to Kinetics.
Dataset: We conduct experiments on ActivityNet-100 (*Caba Heilbron et al.*, 2015) dataset. For this experiment, we provide results when considering 20 action categories with 1500 test videos. We have selected the ActivityNet dataset to make sure there are no same videos in HVU and Kinetics training set. For clustering, the performance is measured using clustering accuracy (*Sharma et al.*, 2019).
Method and Results: We extract features using 3D-ResNet50 and HATNet pre-trained on Kinetics-600 and HVU for the test videos and then cluster them with KMeans clustering algorithm with the given number of action categories. Table 5.11 clearly shows that the features learned using HVU are far more effective compared to features learned using Kinetics.

5.5 Summary

In this chapter, we introduced the Holistic Video Understanding (HVU) dataset, a large-scale multi-task, multi-label video benchmark dataset with comprehensive tasks and annotations. The HVU dataset consists of 572k videos and 9M annotations, covering a wide range of labels including scenes, objects, actions, events, attributes, and concepts. Through our experiments, we demonstrated the potential of the HVU dataset in learning a generic video representation by applying it to three real-world tasks: video classification, video captioning, and video clustering.

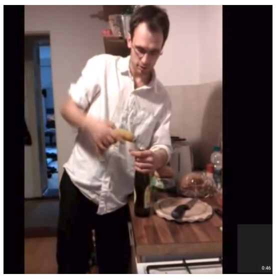
Additionally, we proposed a novel network architecture, HATNet, which combines 2D and 3D

ConvNets to learn a robust spatio-temporal feature representation via multi-task and multi-label learning in an end-to-end manner. Our findings suggest that the HVU dataset and HATNet architecture hold great promise for the advancement of holistic video understanding research.

forest,musician,flutist,music,musical_instrument,brass_instrument,wind_instrument,flautist,recreation,musical_instrument_accessory,plant,playing_flute,tree

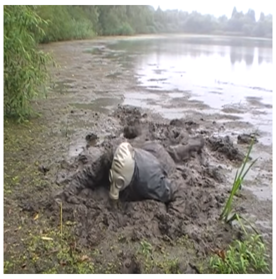
string_instrument,musician,man,,guitarist,plucked_string_instruments,music,tapping_guitar,bass,musical_instrument_accessory,performance,string_instrument_accessory,electric_guitar,sitting,monochrome_photography,musical_instrument,guitar_accessory,resonator

sport_venue,shoe,outdoor_shoe,joint,foot,ball,grass,knee,human_leg,fun,football_player,ball_game,green,footwear,football,player,sports_equipment,juggling_soccer_ball,soccer,plant,soccer_ball,sports,play



opening_bottle_not_wine_,joint,muscle,service,finger,distilled_beverage,fun,taste,standing,arm,t_shirt,glass,alcohol,drink,hand,bottle,photograph,cooking

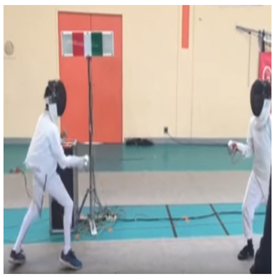
smile,nose,textile,cheek,thigh,mouth,girl,diaper,finger,baby_products,human_leg,fun,playing_xylophone,infant,toy,facial_expression,skin,child,hand,sitting,human_hair_color,daytime,play,toddler



coast,watercourse,plant,wetland,terrain,floodplain,marsh,wading_through_mud,boulder,tree,water,natural_resources,river,rock,waterway,outcrop,shore,creek

Figure 5.7: Video frame samples from HVU with corresponding tags of different categories.

mopping_floor,wood_stain,sleeve,design,man,wood,wood_flooring,gentleman,standing,swab,facial_hair,shirt,outerwear,tartan,flooring,laminate_flooring,floor,dress_shirt,plaid,angle

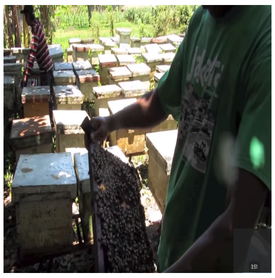


individual_sports,indoor_games_and_sports,joint,games,combat_sport,weapon_combat_sports,leisure,net,fun,recreation,martial_arts,epee,striking_combat_sports,fencing,competition,contact_sport,fencing_sport_,flooring,fencing_weapon,floor,sports,play

italian_food,food,pizza,making_pizza,appetizer,cuisine,pizza_cheese,prosciutto,vegetable,darkness,sicilian_pizza,recipe,rectangle,european_food,flatbread

charcoal,campfire,shovel,smoke,outdoor_grill,fire,animal_source_foods,barbecue_grill,grilling,winter,fun,ice,meat,cooking_on_campfire,roasting,grass,barbequing

hand,multimedia,server,electronics,electronic_device,finger,computer_hardware,technology,assembling_computer,computer_case,arm,magenta,text



bee_keeping,human,grass,backyard,outdoor_structure,wood,tree,forest,human_behavior,beekeeper,leaf,garden,bee,yard,apiary,t_shirt,plant,beehive,male

Figure 5.8: More examples of video frame samples from HVU with corresponding tags of different categories.

CHAPTER 6

Adaptive Spatio-Temporal Convolutional Neural Networks for Efficient Video Classification

This chapter is based on the following publication:
3D CNNs with Adaptive Temporal Feature Resolutions
Mohsen Fayyaz*, Emad Bahrami*, Ali Diba, Mehdi Noroozi, Ehsan Adeli, Luc Van Gool, Juergen Gall
IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2021.
* *denotes equal contribution.*

Below we will summarize the contributions of each author.

- **Mohsen Fayyaz and Emad Bahrami**
  The present contribution is a collaborative effort between Mohsen Fayyaz and Emad Bahrami. Mohsen Fayyaz initially conceptualized the Similarity Guided Sampling (SGS) block and the idea of utilizing 3D CNNs with adaptive temporal feature resolutions, while Emad Bahrami proposed and evaluated various types of similarity measurement approaches and provided the final formulation of the similarity measurement method of SGS. Both authors were also crucial in the implementation and empirical evaluation of the proposed approach. In terms of the preparation of the manuscript, both Mohsen Fayyaz and Emad Bahrami made significant contributions. It is worth highlighting that the authors contributed equally to this work.

- **Ali Diba, Mehdi Noroozi, Ehsan Adeli, and Luc Van Gool**
  The authors made substantial contributions to this work through their ideas, discussions, and writing. Additionally, Mehdi Noroozi and Ehsan Adeli also provided valuable computing resources.

- **Juergen Gall**
  This work was made possible through the supervision and funding provided by Juergen Gall. In addition to his financial and managerial support, the author also contributed through suggestions, discussions, and writing.

Contents

In previous chapters, we explored various approaches to enhance model performance in video understanding by introducing innovative spatio-temporal modeling techniques, novel knowledge transfer methods, and state-of-the-art architectures. However, the high computational cost and GFLOP requirements often limit the effectiveness of advanced 3D Convolutional Neural Networks (CNNs). To address this challenge, we propose a novel approach to reduce the computational complexity of 3D CNNs. In this chapter, we introduce a differentiable Similarity Guided Sampling (SGS) module that can seamlessly integrate into existing 3D CNN architectures, enabling the utilization of adaptive temporal feature resolutions (ATFR). The SGS module is designed to learn the similarity of temporal features and group them accordingly, resulting in a variable temporal feature resolution optimized for each specific input video clip. By incorporating the SGS module into 3D CNNs, we can substantially decrease their computational requirements while maintaining or even enhancing accuracy. To demonstrate the effectiveness of our proposed SGS module, we conducted evaluations on various datasets, including Kinetics-600, Kinetics-400, Mini-Kinetics, Something-Something V2, UCF101, and HMDB51. By integrating the SGS module into multiple state-of-the-art 3D CNNs, we achieved an average reduction in GFLOPs of approximately 50% while preserving high performance across these diverse datasets.

6.1 Introduction

In recent years, there has been tremendous progress in video processing in the light of new and complex deep learning architectures, which are based on variants of 3D Convolutional Neural Networks (CNNs) (*Tran et al.*, 2015; *Feichtenhofer et al.*, 2019; *Diba et al.*, 2017, 2018, 2019; *Tran et al.*,

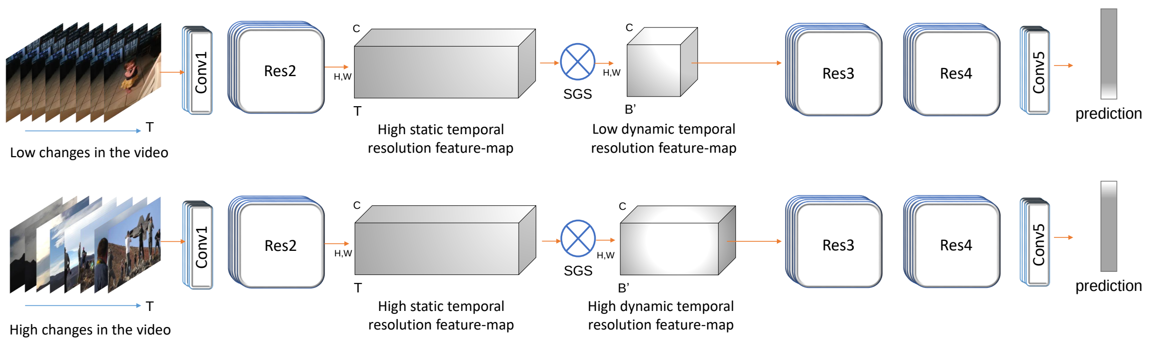

Conv1
Res2
H,W
C
T
High static temporal resolution feature-map
SGS
H,W
C
B'
Low dynamic temporal resolution feature-map
Res3
Res4
Conv5
prediction
T
Low changes in the video
Conv1
Res2
H,W
C
T
High static temporal resolution feature-map
SGS
H,W
C
B'
High dynamic temporal resolution feature-map
Res3
Res4
Conv5
prediction
T
High changes in the video

Figure 6.1: The difficulty of recognizing actions varies largely across videos. For videos with slow motion (top), the temporal features that are processed within a 3D CNN can be highly redundant. However, there are also very challenging videos where all features are required to understand the content (bottom). While previous 3D CNNs use fix down-sampling schemes that are independent of the input video, we propose a similarity guided sampler that groups and aggregates redundant information of temporal features into $B' \leqslant T$ feature maps. The core aspect is that this process adapts the internal temporal resolution to the input video such that B' is small if the input features are redundant (top) and large (bottom) if most of the features are required.

2017; *Feichtenhofer*, 2020). They are trained for a specific number of input frames, typically between 16 to 64 frames. For classifying a longer video, they slide over the video and the outputs are then aggregated. These networks, however, are often costly to train and heavy to deploy for inference tasks. In order to reduce the inference time, *Korbar et al.* (2019) and *Zhu et al.* (2020) proposed to process not all parts of a video with the same 3D CNN. While *Korbar et al.* (2019) train a second network that decides for each chunk of input frames if it should be processed by the more expensive 3D CNN, *Zhu et al.* (2020) use a fixed scheme where a subset of the input chunks are processed by an expensive 3D CNN and the other chunks by a less expensive 3D CNN. The latter then uses an RNN to fuse the outputs of the different 3D CNNs. Although both approaches effectively reduce the GFLOPS during inference, they increase the training time since two instead of one network need to be trained. Furthermore, they do not reduce the computational cost of the 3D CNNs themselves.

In this work, we propose an approach that makes 3D CNNs more efficient for training and inference. Our proposal is based on the observation that the computational cost of a 3D CNN depends on the temporal resolution it operates on at each stage of the network. While the temporal resolution can be different at different stages, the schemes that define how the temporal resolution is reduced are hard-coded and thus the same for all videos. However, it is impossible to define a scheme that is optimal for all videos. If the temporal resolution is too much reduced, the network is forced to discard important information for some videos. This results in a decrease in the action recognition accuracy performance. Vice versa, a high temporal resolution results in highly redundant feature maps and increases the computational time, which makes the 3D CNN highly inefficient for most videos. In this work, we, therefore, address the question of how a 3D CNN can dynamically adapt its computational resources in a way such that not more resources than necessary are used for each input chunk.

In order to address this question, we propose to exploit the redundancy within temporal features such that 3D CNNs process and select the most valuable and informative temporal features for the action classification task. In contrast to previous works, we propose to dynamically adapt the temporal feature resolution within the network to the input frames such that on one hand important information is not discarded and on the other hand no computational resources are wasted for processing redundant information. To this end, we propose a *Similarity Guided Sampling (SGS)* mechanism that measures the similarity of temporal feature maps, groups similar feature maps together, and aggregates the grouped feature maps into a single output feature map. The similarity guided sampling is designed such that it is differentiable and number of output feature maps varies depending on the redundancy of the temporal input feature maps as shown in Fig. 6.1. By integrating the similarity guided sampling as an additional module within any 3D CNN, we convert the 3D CNN with fixed temporal feature resolutions into a much more efficient dynamic 3D CNN with *adaptive temporal feature resolutions (ATFR)*. Note that this approach is complementary to the works by *Korbar et al.* (2019) and *Zhu et al.* (2020), and the two static 3D CNNs used in these works can be replaced by adaptive 3D CNNs. However, even with just a single 3D CNN with adaptive temporal feature resolutions, we already achieve a higher accuracy and lower GFLOPs performance compared to the works by *Korbar et al.* (2019) and *Zhu et al.* (2020).

We demonstrate the efficiency of 3D CNNs with adaptive temporal feature resolutions by integrating the similarity guided sampler into the current state-of-the-art 3D CNNs, such as R(2+1)D (*Tran et al.*, 2018b), I3D (*Carreira and Zisserman*, 2017), and X3D (*Feichtenhofer*, 2020). Our approach drastically decreases the GFLOPs by about half on average while the accuracy remains nearly the same or gains improvements. In summary, the similarity guided sampler is capable of significantly scaling down the computational cost of off-the-shelf 3D CNNs and therefore plays a crucial role in real-world video-based applications.

6.2 Adaptive Temporal Feature Resolutions

Current state-of-the-art 3D CNNs operate at a static temporal resolution at all levels of the network. Due to the redundancy of neighboring frames, traditional 3D CNN methods often down-sample the temporal resolution inside the network. This helps the model to operate at a lower temporal resolution and hence reduces the computation cost. The down-sampling, however, is static which has disadvantages in two ways. First, a fixed down-sampling rate can discard important information, in particular for videos with very fast motion as is for instance the case for ice hockey games. Second, a fixed down-sampling rate might still include many redundant temporal features that do not contribute to the classification accuracy as it is for instance the case for a video showing a stretching exercise. We, therefore, propose a module that dynamically adapts the temporal feature resolution within the network to the input video such that on one hand important information is not discarded and on the other hand no computational resources are wasted for processing redundant information.

Fig. 6.1 illustrates a 3D CNN with *adaptive temporal feature resolutions (ATFR)*. The core aspect of ATFR is to fuse redundant information from a temporal sequence of features and extract only the most relevant information in order to reduce the computational cost for processing a video. An important aspect is that this approach is not static, *i.e.*, the amount of information that is extracted varies for each video as illustrated in Fig. 6.1. In order to achieve this, we propose a novel *Similarity Guided Sampling (SGS)* mechanism that will be described in Sec. 6.3.

In principle, any 3D CNN can be converted into a CNN with adaptive temporal feature resolutions

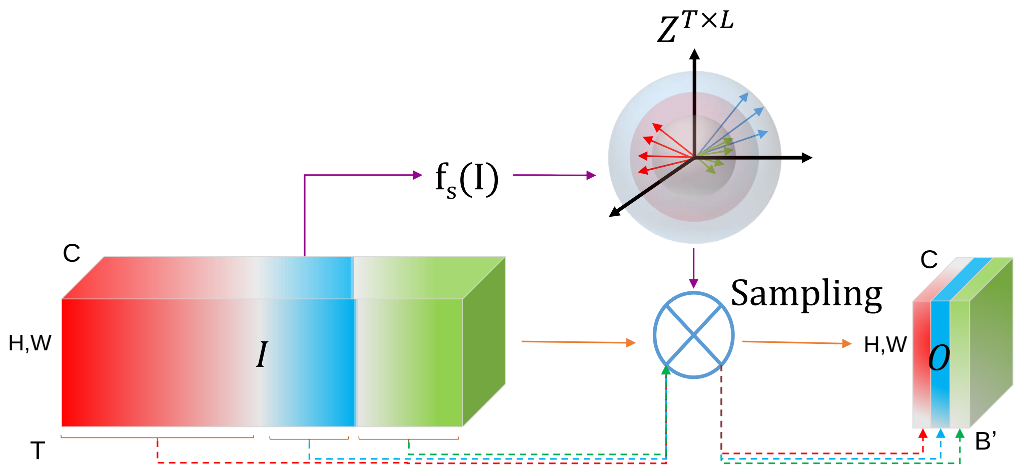


Figure 6.2: To learn the similarity of the feature maps, we map each temporal feature map $\mathcal{I}_t$ using $f_s(\mathcal{I})$ into an L-dimensional similarity space. After the mapping, $\mathcal{Z} \in \mathbb{R}^{T\times L}$ contains all of the feature maps represented as vectors in the similarity space. Afterward, we group similar vectors by creating B similarity bins. Using the similarity bins, the sampler aggregates the similar feature maps of each bin into the output feature map $\mathcal{O}_b$.

by using our SGS module. Since the module is designed to control the temporal resolution within the network for each video, it should be added to the early stages of a network in order to get the best reduction of computational cost. For R(2+1)D (*Tran et al.*, 2018b), for example, we recommend to add SGS after the second ResNet block. This means that the temporal resolution is constant for all videos before SGS, but it dynamically changes after SGS. We discuss different 3D CNNs with SGS in Sec. 6.4.1.

6.3 Similarity Guided Sampling

The *SGS* is a differentiable module to sample spatially similar feature maps over the temporal dimension and aggregate them into one feature map. Since the number of output feature maps is usually lower than the input feature maps, i.e., $B' < T$, redundant information is removed as illustrated in Figure 6.2. The important aspect is that B' is not constant, but it varies for each video. In this way, we do not remove any information if there is no redundancy among the input feature maps.

This means that we need to a) learn the similarity of feature maps, b) group similar feature maps, and c) aggregate the grouped feature maps. Furthermore, all these operations need to be differentiable. We denote an input feature map for frame t by $\mathcal{I}_t \in \mathbb{R}^{C\times H\times W}$, where C, H, and W denote the number of channels, height, and width, respectively. To learn the similarity of the feature maps, we map each feature map $\mathcal{I}_t$ into an L-dimensional similarity space. This mapping $f_s(\mathcal{I}_t)$ is described in Sec. 6.3.1. After the mapping, $\mathcal{Z} \in \mathbb{R}^{T\times L}$ contains all feature maps in the similarity space, which are then grouped and aggregated into B' feature maps. The grouping of $\mathcal{Z}_t$ is described in Sec. 6.3.2 and the aggregation of the grouped features in Sec. 6.3.3.

6.3.1 Similarity Space

The similarity space is a L dimensional vector space where each temporal input feature map is represented by a vector $\mathcal{Z}_t$. The mapping is performed by the similarity network $f_s(\mathcal{I})$ that consists of a global average pooling layer and two convolutional layers. The pooling is applied over the spatial dimension of the feature map while keeping the temporal dimension. Afterward two $1D$ convolutional layers are applied with kernel sizes of 1 and output channel sizes C and L, respectively.

6.3.2 Similarity Bins

To group similar feature maps $\mathcal{I}_t$, we use the magnitude of each vector $\mathcal{Z}_t$, i.e.,

$$\Delta_t = ||\mathcal{Z}_t|| \tag{6.1}$$

and we consider two feature maps $\mathcal{I}_t$ and $\mathcal{I}_{t'}$ similar if the value of Δ_t and $\Delta_{t'}$ lie inside a similarity bin. To make the grouping very efficient and differentiable, we propose a binning approach with B similarity bins. We set $B = T$ such that no information is discarded if there is no redundancy between the feature maps of all frames. For most videos, a subset of bins remain empty and will be discarded such that the remaining bins, B', will be less than B as it is described in Sec. 6.3.3.

We first estimate the half of the width of each similarity bin γ, by computing the maximum magnitude Δ_{max} and dividing it by the number of the desired bins B:

$$\Delta_{max} = \max(\Delta_1, \dots, \Delta_T), \;\; \gamma = \frac{\Delta_{max}}{2B}. \tag{6.2}$$

Having the width of the similarity bins, the center of each bin β_b is estimated as follows:

$$\beta_b = (2b-1)\gamma \quad \forall b \in (1, \dots, B). \tag{6.3}$$

6.3.3 Differentiable Bins Sampling

The grouping and aggregation of all feature maps $\mathcal{I}_t$ based on the bins B will be done jointly by sampling temporal feature maps which belong to the same similarity bin and add them together. We denote the aggregated feature maps for each bin b by $\mathcal{O}_b \in \mathbb{R}^{C\times H\times W}$. To make the process differentiable, we use generic differentiable sampling kernels $\Psi(., \beta_b)$ that are defined such that a sampler only samples from the input temporal feature map $\mathcal{I}_t$ if Δ_t lies in the similarity bin b. This can be written as:

$$\mathcal{O}_b = \sum_{t=1}^{T} \mathcal{I}_t \Psi(\Delta_t, \beta_b). \tag{6.4}$$

Theoretically, any differentiable sampling kernel that has defined gradients or sub-gradients with respect to Δ_t can be used. In our experiments, we evaluate two sampling kernels. The first kernel is based on the Kronecker-Delta function δ:

$$\mathcal{O}_b = \frac{1}{\sum_{t=1}^{T} \delta\left(\left\lfloor \frac{|\Delta_t - \beta_b|}{\gamma} \right\rfloor\right)} \sum_{t=1}^{T} \mathcal{I}_t \delta\left(\left\lfloor \frac{|\Delta_t - \beta_b|}{\gamma} \right\rfloor\right). \tag{6.5}$$

The kernel averages the feature maps that end in the same bin. As second kernel, we use a linear sampling kernel:

$$\mathcal{O}_b = \sum_{t=1}^{T} \mathcal{I}_t \max\left(0, 1 - \frac{|\Delta_t - \beta_b|}{\gamma}\right). \tag{6.6}$$

The kernel gives a higher weight to feature maps that are closer to β_b and less weights to feature maps that are at the boundary of a bin. While we evaluate both kernels, we use the linear kernel by default.

After the sampling, some bins remain empty, i.e., $\mathcal{O}_b = 0$. We drop the empty bins and denote by B' the bins that remain. Note that B' varies for each video as illustrated in Fig. 6.1. In our experiments we show that the similarity guided sampling can reduce the GFLOPS of a 3D CNN by over 47% in average, making 3D CNNs suitable for applications where they are computationally expensive.

6.3.4 Backpropagation

Using differentiable kernels for sampling, gradients can be backpropagated through both $\mathcal{O}$ and Δ, where Δ is the magnitude of the similarity vectors $\mathcal{Z}$ which are the outputs of $f_s(.)$. Therefore, we can backpropagate through $f_s(.)$. For the linear kernel (6.6), which we use if not otherwise specified, the gradient with respect to $\mathcal{I}_t$ is given by

$$\frac{\partial \mathcal{O}_b}{\partial \mathcal{I}_t} = \max\left(0, 1 - \frac{|\Delta_t - \beta_b|}{\gamma}\right) \tag{6.7}$$

and the gradient with respect to Δ_t is given by

$$\frac{\partial \mathcal{O}_b}{\partial \Delta_t} = \mathcal{I}_t \begin{cases} 0 & |\beta_b - \Delta_t| \geqslant \gamma \\ \frac{1}{\gamma} & \beta_b - \gamma < \Delta_t \leqslant \beta_b \\ -\frac{1}{\gamma} & \beta_b < \Delta_t < \beta_b + \gamma \end{cases}. \tag{6.8}$$

Note that for computing the sub-gradients (6.8) only the kernel support region for each output bin needs to be considered. The sampling mechanism can therefore be efficiently implemented on GPUs.

6.4 Experiments

We evaluate our proposed method on the action recognition benchmarks Mini-Kinetics (*Xie et al.*, 2018b), Kinetics-400 (*Kay et al.*, 2017), Kinetics-600 (*Carreira et al.*, 2018), Something-Something-V2 (*Goyal et al.*, 2017b), UCF-101 (*Soomro et al.*, 2012), and HMDB-51 (*Kuehne et al.*, 2011). For these datasets, we use the standard training/testing splits and protocols provided by the datasets.

6.4.1 Implementation Details

3D CNNs with ATFR. Similarity guided sampling (SGS) is a differentiable module that can be easily implemented in current deep learning frameworks. We have implemented our SGS module as a new layer in PyTorch which can be easily added to any 3D CNN architecture. To better evaluate the SGS, we have added it to various backbones, such as R(2+1)D (*Tran et al.*, 2018b), I3D (*Carreira and

Zisserman, 2017), X3D (*Feichtenhofer*, 2020), and a modified 3DResNet. We place our SGS layer on the second stage of the backbone models. For all of the X3D based models, we follow the training, testing, and measurement setting by *Feichtenhofer* (2020) unless mentioned otherwise. Additional details and codes are available online.[1]

Training. Our models on Mini-Kinetics, Kinetics-400, and Kinetics-600 are trained from scratch using randomly initialized weights without any pre-training. However, we fine-tune on Something-Something-V2, UCF-101, and HMDB-51 with models pre-trained on Kinetics-400. We trained our models using SGD with momentum 0.9 and a weight decay of 0.0001 following the setting of *Feichtenhofer et al.* (2019). For Kinetics and Mini-Kinetics, we use a half-period cosine schedule (*Loshchilov and Hutter*, 2017) with a linear warm-up strategy (*Goyal et al.*, 2017a) to adapt the learning rate over 196 epochs of training. During training, we randomly sample 32 frames from a video with input stride 2. For spatial transformations, we first scale the shorter side of each frame with a random integer from the interval between 256 and 320 (*Wang et al.*, 2018c; *Feichtenhofer et al.*, 2019; *Simonyan and Zisserman*, 2015) then we apply a random cropping with size 224×224 to each frame. Furthermore, each frame is horizontally flipped with the probability of 0.5.

Testing. We follow *Wang et al.* (2018c) and *Feichtenhofer et al.* (2019), and uniformly sample 10 clips from each video for inference. The shorter side of each clip is resized to 256 and we extract 3 random crops of size 256×256 from each clip. For the final prediction, we average the softmax scores of all clips.

Measurements. We report top-1 and top-5 accuracy. To measure the computational efficiency of our models, we report the complexity of the models in GFLOPS based on a single input video sequence of 32 frames and spatial size 224×224 for validation and 256×256 for testing. As shown in Fig. 6.3, 3D CNNs with ATFR adapt the temporal feature resolutions and the GFLOPs vary for different clips. For ATFR models, we therefore report the average GFLOPs.

6.4.2 Ablation Experiments

We first analyze different setups for our SGS module. Then, we analyze the efficiency and effect of using our SGS module in different 3D CNN models. If not otherwise specified, we use 3DResNet-18 as 3D CNNs backbones and report the results on the Mini-Kinetics validation set.

6.4.2.1 Different Similarity Measurements

As mentioned in Sec. 6.3.2, we use the magnitude of the embedding vectors as the similarity measurement to create the similarity bins. The embedding vectors are represented in an L dimensional space. Instead of magnitudes, we can use other measures such as the directions of the vectors. To better study this, we convert the Cartesian coordinates of the vectors to spherical coordinates. In an L dimensional space, a vector is represented by 1 radial coordinate and $L - 1$ angular coordinates. To use the spherical coordinates of the vectors for creating the similarity bins, we use multi-dimensional bins and sampling kernels.

We report the results in Table 6.1. As can be seen, using the magnitudes of the vectors results in a better accuracy compared to angular coordinates or spherical coordinates. We believe that due to the similarity of the neighboring video frames using only magnitudes of the vectors for the similarity

[1] `https://SimilarityGuidedSampling.github.io`

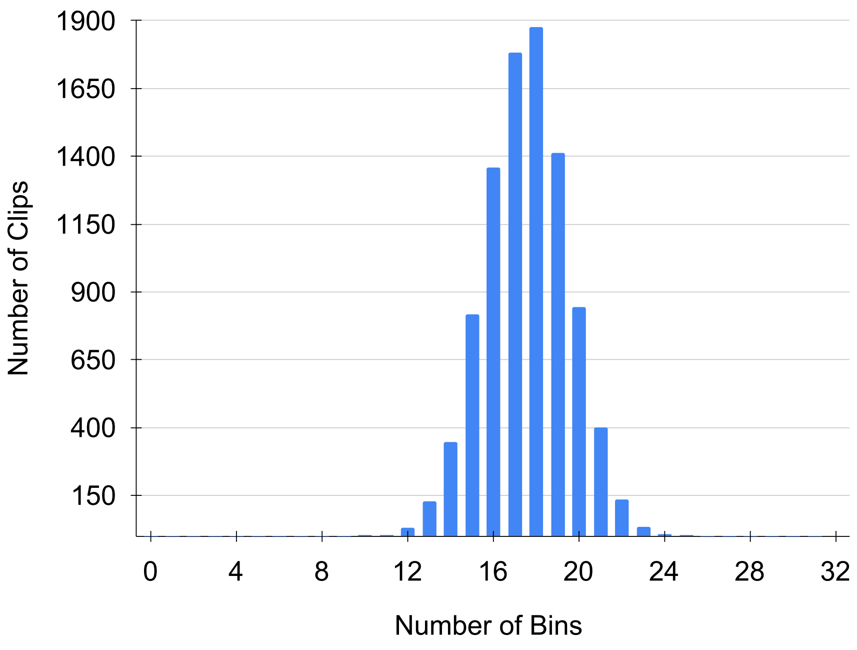


Figure 6.3: Histogram of active bins for 3DResNet-50 + ATFR on the Mini-Kinetics validation set. The y-axis corresponds to the number of clips and the x-axis to the number of active bins B'.

Similarity	Magnitude	Angular	Spherical
top1	**69.6**	68.5	68.7
top5	**88.8**	87.8	88.1

Table 6.1: Impact of the similarity measure for 3DResNet-18 + ATFR on Mini-Kinetics with linear sampling kernel. We show top-1 and top-5 classification accuracy (%).

measurement is enough and angular or spherical coordinates add too much of complexity to the model. In all of the experiments, the number of bins B is equal to 32. For the angular coordinates, we divide the angles into 4 and 8 bins (4×8). For the spherical coordinates, we divide the radial coordinate into 2 and the angular coordinates into 4 and 4 bins $(2 \times 4 \times 4)$.

6.4.2.2 Different Sampling Kernels

As mentioned in Sec. 6.3.3, we can use different differentiable sampling kernels (6.4). We evaluate two different sampling kernels, namely the Kronecker-Delta sampling kernel (6.5) and the linear sampling kernel (6.6). As can be seen in Table 6.2, the linear kernel performs better than the Kronecker-Delta kernel. The slight superiority of the linear kernel is due to the higher weighting of the temporal feature maps that are closer to the center of the bins. We use the linear kernel for the rest of the paper.

Kernel	Linear	Kronecker
top1	**69.6**	68.9
top5	**88.8**	88.6

Table 6.2: Impact of the sampling kernel.

L	1	4	8	16
top1	67.3	68.4	**69.6**	64.7
top5	87.7	88.1	**88.8**	86.1

Table 6.3: Impact of the dimensionality L of the similarity space.

6.4.2.3 Embedding Dimension

As mentioned in Sec. 6.3.1, we map the temporal feature maps into an L-dimensional similarity space. In Table 6.3, we quantify the effect of L. The accuracy increases as L increases until $L = 8$. For $L = 16$ the dimensionality is too large and the similarity space tends to overfit. The model with $L = 1$ is a special case since it can be considered as a direct prediction of Δ_t (6.1) without mapping the temporal features into a similarity space $\mathcal{Z}_t$. The results show that using a one dimensional embedding space results in a lower accuracy, which demonstrates the benefit of the similarity space.

6.4.2.4 Different Input Frame-rates

It is an interesting question to ask how a 3D CNN with ATFR performs when the number of input frames or the stride changes for inference. To answer this question, we train two 3D CNNs with ATFR and two without ATFR using 32 input frames and a sampling stride of 2, which corresponds to a temporal receptive field of 64 frames. For inference, we then change the number of frames to 64 and/or the stride to 1.

As it can be seen in Table 6.4, increasing the input frames from 32 to 64 improves the accuracy of all models. This improvement in accuracy is due to the increase in the temporal receptive field over the input frames while keeping the temporal input resolution. However, the computation cost of the models without ATFR increases as expected by factor 2. If ATFR is used, the increase is only by 1.7 and 1.5 for R(2+1)D+ATFR and 3DResNet-18+ATFR. By comparing R(2+1)D with R(2+1)D+ATFR, we see how ATFR drastically reduces the GFLOPS from 46.5 to 32.3 for 32 frames and from 93.1 to 54.9 for 64 frames. This shows that more frames also increase the redundancy and ATFR efficiently discards this redundancy. Furthermore, it demonstrates that ATFR is robust to changes of the frame-rate and number of input frames.

It is also interesting to compare the results for 32 frames with stride 2 to the results for 64 frames with stride 1. In both cases, the temporal receptive field is 64. We can see the efficiency of our method in adapting the temporal resolution compared to the traditional static frame-rate sampling methods, *i.e.*, 3DResNet-18+ATFR operates on average with 21.1 GFLOPs for 64 input frames compared to SlowFast with GFLOPs of 30.9 (32) and 61.8 (64), and R(2+1)D with GFLOPs of 46.5 (32) and 93.1 (64).

model	input frames	GFLOPs	top1		top5	
			stride			
			1	2	1	2
SlowFast-8x8-ResNet18	32	30.9	67.5	69.7	87.1	89.1
	64	61.8 (2.0)	72.1	74.6	89.9	91.9
R(2+1)D	32	46.5	67.4	69.3	86.2	87.5
	64	93.1 (2.0)	70.8	73.7	88.8	91.5
R(2+1)D+ATFR	32	32.3	67.4	69.3	86.4	87.6
	64	54.9 (1.7)	71.4	73.8	88.6	90.7
3DResNet-18+ATFR	32	**14.0**	67.3	69.6	87.1	88.8
	64	**21.1 (1.5)**	72.1	74.8	89.8	91.4

Table 6.4: Impact of the stride and number of input frames during inference. All models are trained with 32 frames and stride 2.

Lowest Temporal Resolution	Highest Temporal Resolution
presenting weather forecast	passing American football (in game)
stretching leg	swimming breast stroke
playing didgeridoo	playing ice hockey
playing clarinet	pushing cart
golf putting	gymnastics tumbling

Table 6.5: The 5 action classes with lowest and highest required adaptive temporal resolution for 3DResNet-50 + ATFR on Mini-Kinetics.

6.4.2.5 Adaptive Temporal Feature Resolutions

As shown in Fig. 6.3, the temporal feature resolutions vary for different clips. In order to analyze how the temporal feature resolution relates to the content of a video, we report in Table 6.5 the 5 action classes with lowest adaptive temporal feature resolution (<12) and highest adaptive temporal feature resolution (>20). As in Fig. 6.3, the results are for the 3DResNet-50+ATFR on the Mini-Kinetics validation set. As it can be seen, the actions with less movements like 'presenting weather forecast' result in a low temporal resolution while actions with fast (camera) motions like 'passing American football (in game)' result in a high temporal resolution.

6.4.2.6 SGS Placement

To evaluate the effect of the location of our SGS module within a 3D CNN, we add it to different stages of X3D-S (*Feichtenhofer*, 2020) and train it on Mini-Kinetics. As it can be seen in Table 6.6, adding SGS to the first stage of X3D-S drastically reduces the GFLOPs by 52.6% (2.1×) while getting slightly lower accuracy. On the other hand, adding SGS after the 2^{nd} stage results in a 42.1% reduction of GFLOPs and slightly higher accuracy. The same accuracy and growth in GFLOPs occurs when SGS is added after the 3^{rd} stage.

Stage	No SGS	First Conv	Res2	Res3
top1	77.9	77.8	**78.0**	78.0
GFLOPs	1.9	**0.9**	1.1	1.3

Table 6.6: Evaluating the result of adding our SGS layer to different stages of a X3D-S network on Mini-Kinetics.

model	backbone	GFLOPs	top1	top5
Fast-S3D *Xie et al.* (2018b)	-	43.5	78.0	-
SlowFast 8x8	ResNet18	40.4	77.5	93.3
SlowFast 8x8	ResNet50	65.7	79.3	94.2
R(2+1)D	ResNet50	101.8	78.7	93.4
R(2+1)D+**ATFR**	ResNet50	67.3	78.2	92.9
I3D	ResNet50	148.4	79.3	94.4
I3D+**ATFR**	ResNet50	105.2	78.8	93.6
X3D-S	-	1.9	77.9	93.4
X3D-S+**ATFR**	-	**1.1**	78.0	93.5
3DResNet	ResNet50	40.8	79.2	94.6
3DResNet+**ATFR**	ResNet50	**23.4**	79.3	94.6

Table 6.7: Comparison with state-of-the-art methods on Mini-Kinetics. The accuracy for Fast-S3D (*Xie et al.*, 2018b) is reported with 64 frames.

6.4.3 Mini-Kinetics

Mini-Kinetics is a smaller dataset compared to the full Kinetics-400 dataset (*Kay et al.*, 2017) and consists of 200 categories. Since some videos on YouTube are not accessible, the training and validation set contain 144,132 and 9182 video clips, respectively. Table 6.7 shows the results on Mini-Kinetics. We add the SGS module to four 3D CNNs R(2+1)D (*Tran et al.*, 2018b), I3D (*Carreira and Zisserman*, 2017), X3D (*Feichtenhofer*, 2020), and 3DResNet. In all cases, ATFR drastically reduces the GFLOPS while the accuracy remains nearly the same. For X3D, the accuracy even increases marginally.

6.4.4 Kinetics-400

We also evaluate ATFR with state-of-the-art 3D CNNs on Kinetics-400 (*Kay et al.*, 2017), which contains $\sim$240k training and $\sim$20k validation videos of 400 human action categories. Table 6.8 shows the comparison with the state-of-the-art. We add the SGS module to the state-of-the-art 3D CNNs SlowFast (*Feichtenhofer et al.*, 2019) and three versions of X3D (*Feichtenhofer*, 2020).

As it can be seen, our SGS module drastically decreases the GFLOPs of all 3D CNNs. In contrast to Mini-Kinetics, it even improves the accuracy for all 3D CNNs. We will see that this is the case for all large datasets. For X3D-XL (*Feichtenhofer*, 2020), we observe a $\sim$45% reduction in GFLOPs and 0.2% improvement in accuracy. We can see that X3D-XL+ATFR$^{\beta}$ requires similar GFLOPs compared to X3D-L$^{\beta}$ (*Feichtenhofer*, 2020) while providing a higher accuracy by 1.8%. We can also see that X3D-XL+ATFR$^{\alpha}$ requires drastically less GFLOPs compared to X3D-L$^{\beta}$ (*Feichten-*

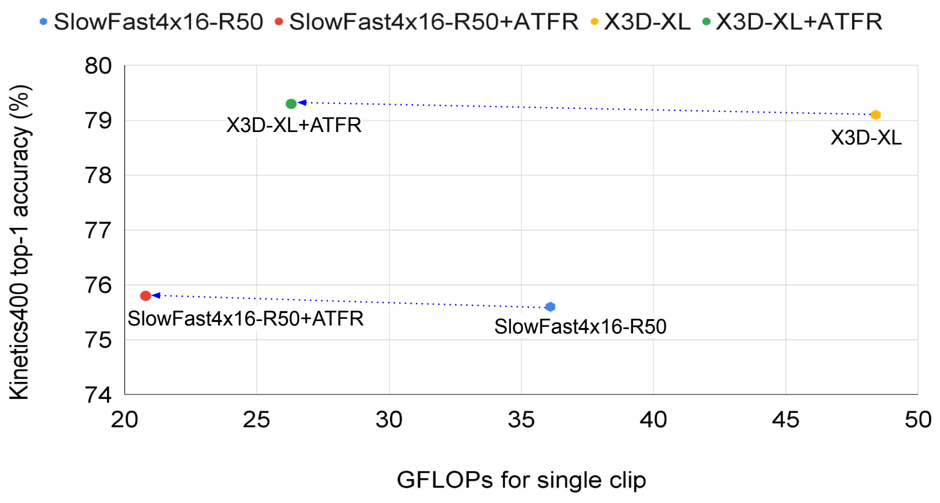


Figure 6.4: Accuracy vs. GFLOPs for the Kinetics-400 validation set.

hofer, 2020) while getting a higher accuracy by 1.1%. In comparison to the computational heavy SlowFast16×8,R101+NL (*Feichtenhofer et al.*, 2019), X3D-XL+ATFR$^{\beta}$ gets comparable top-1 accuracy while having 8.9× less GFLOPs.

Comparing the 3D CNNs with ATFR to SCSampler (*Korbar et al.*, 2019) and FASTER (*Zhu et al.*, 2020), which require training two networks, our approach with a single adaptive 3D CNN achieves a higher accuracy and lower GFLOPs. Note that our approach is complementary to the methods by *Korbar et al.* (2019) and *Zhu et al.* (2020), and the two static 3D CNNs used in these works can be replaced by adaptive 3D CNNs. Nevertheless, our approach outperforms these works already with a single 3D CNN. Fig. 6.4 shows the accuracy/GFLOPs trade-off for a few 3D CNNs with and without ATFR.

6.4.5 Kinetics-600

We also evaluate our approach on the Kinetics-600 dataset (*Carreira et al.*, 2018). As shown in Table 6.9, ATFR shows a similar performance as on Kinetics-400. Our SGS module drastically decreases the GFLOPs of all 3D CNNs while improving their accuracy. For X3D-XL (*Feichtenhofer*, 2020), we observe a ∼47.1% reduction of GFLOPs and a slight improvement in accuracy. The best model X3D-XL+ATFR achieves state-of-the-art accuracy. Note that the average GFLOPs of X3D-XL+ATFR are even lower on Kinetics-600 compared to Kinetics-400. This shows that the additional videos of Kinetics-600 are less challenging in terms of motion, which is also reflected by the higher classification accuracy. Compared to SlowFast16×8, R101+NL (*Feichtenhofer et al.*, 2019), it requires about 9× less GFLOPs.

6.4.6 Something-Something-V2

We finally provide results for the Something-Something V2 dataset (*Goyal et al.*, 2017b). It contains 169K training and 25K validation videos of 174 action classes that require more temporal modeling compared to Kinetics. Following *Wu et al.* (2020), we use a R50-SlowFast model pre-trained on

Kinetics-400 with 64 frames for the fast pathway, the speed ratio of $\alpha = 4$, and channel ratio $\beta = 1/8$. Similar to Kinetics, the SGS module reduces the GFLOPs by 33.9% while keeping the accuracy almost the same.

6.4.7 UCF101 and HMDB51 Results

UCF-101 (*Soomro et al.*, 2012) contains 13K videos with 101 action classes. It is split into 3 splits with around 9.5K videos in each. For this dataset, we report the average accuracy over three splits.

HMDB-51 (*Kuehne et al.*, 2011) has about 7000 videos with 51 action classes. It contains 3 splits for training and validation. Similar to UCF-101, we report the average accuracy over three splits. Table 6.11 shows the results on UCF-101 and HMDB51. The GFLOPs of our 3DResNet-R50+ATFR on UCF-101 and HMDB-51 are 22.2 and 23.1, respectively. As it can be seen, 3DResNet+ATFR gets comparable results compared to other 3D CNNs while having less GFLOPs as discussed in the paper.

6.4.8 Implementation Details

Modified 3DResNet-18 The architecture of our modified 3DResNet-18 is shown in Table 6.12. In the case of 3DResNet-18+ATFR, we place SGS after the *ResBlock 2*.

SlowFast-8x8-R50+ATFR Following *Wu et al.* (2020) for training on the Something-Something-V2 dataset, the input temporal length to the SlowFast-8x8-R50+ATFR is set to 64. Due to the intensive size of the temporal domain, we limit the the temporal domain size of the SGS for each path-way. For the fast path-way we set the temporal domain size to 8. In other words, SGS is applied over temporal blocks with temporal length of 8 and temporal stride of 8. For the slow path-way we set the temporal domain size to 2. Since we drop zero bins in SGS, this may cause size mismatch for fusion in lateral connections. We therefore zero pad the smaller size tensors to the bigger ones.

6.4.9 Runtime

To evaluate the runtime, we use X3D-S as the base model and report the runtimes for training and inference. As shown in Table 6.13, SGS reduces the training time on Kinetics by 10h. The ATFR equipped model processes almost 51% more frames per second (fps) during inference. Our approach also requires less memory and we can use a larger batch size (BS), namely 256 instead of 208. This shows that the proposed approach substantially reduces GFLOPs, training and inference time, and memory usage.

6.4.10 Different number of bins

The number of sampling bins B controls the maximum number of possible output feature maps of the SGS module. By setting $B = T$, the SGS module can keep all feature maps in case it is needed. To study the effect of changing B, we have evaluated the model performance by changing B during training and inference. The base model is the 3DResNet-18 (Fig. 6.12) trained on Mini-Kinetics. As it can be seen in Table 6.14, reducing B decreases the accuracy, but also the GFLOPS. This is expected since SGS is forced to discard information for each video if $B < T$ even if there is no redundancy among the feature maps.

As a second experiment, we change the number of bins only for inference while we train the model with $B = 32$. This setting is interesting since it shows how flexible the approach is and if

GFLOPS can be reduced at inference time without the need to retrain the model. The results are shown in Table 6.15. If we compare the results with Table 6.14, we observe that the accuracy for training with $B = 32$ and testing with $B = 16$ is only slightly lower than training and testing with $B = 16$. This shows that the GFLOPS can be reduced on the fly if it is required. However, if the difference between the number of bins during training and during inference is getting larger, the accuracy drops.

6.4.11 Cartesian/Spherical Coordinates

As mentioned in Sec. 5.2.1, we use the magnitude of the embedding vectors as the similarity measurement to create the similarity bins. Instead of magnitudes, we can use other measures. While the results are reported in Table 1, we describe how the approach works with spherical coordinates.

To use the spherical coordinates of the vectors for creating the similarity bins, we use multi-dimensional bins and sampling kernels. In an L dimensional spherical coordinate system, we can use different subsets of coordinates for Δ_t^k with the varying number of elements K to create similarity bins, *e.g.*, $K = L$ when using all of the coordinates, $K = L - 1$ when using angular coordinates, or $K = 1$ when using the radial coordinate only. Therefore, similar to Eq. (2) and (3) of the paper, we can estimate β_b^k for every Δ^k.

By using a sampling kernel $\Psi(\Delta_t^k, \beta_b^k)$ as in Eq. (4) of the paper but for each k, a differentiable multi-dimensional sampling operation can be defined by

$$\mathcal{O}_b = \sum_{t=1}^{T} \mathcal{I}_t \prod_{k=1}^{K} \Psi(\Delta_t^k, \beta_b^k). \tag{6.9}$$

6.4.12 Similarity Guided Sampling Visualization

The SGS layer aggregates similar input temporal feature maps into the same output feature map. To better understand such aggregation operation, we have visualized the input and output feature maps of the SGS layer in Figure 6.5. We have used a 3DResNet-50+ATFR trained on the Mini-Kinetics dataset. The sampling kernel used in this experiment is the linear kernel and the number of bins is set to 32. As it can be seen in Figure 6.5, the input temporal feature maps are aggregated to 4 output feature maps. The aggregated feature maps contain both the spatial and temporal information. In this example, the 4^{th} channel of the aggregated feature maps capture some motion flow that can be seen in the visualization.

6.4.13 Comparison to Attention/Gating Mechanisms

To better analyze the effect of our similarity guided sampling mechanism, we add attention modules to the base model and compare the final accuracy and GFLOPs to the base model and the ATFR model. To this end, we use a temporal attention mechanism following *Chen et al.* (2018). Similar to our SGS module, we add this attention module on top of the ResBlock2. As it can be seen in Table 6.16, the model equipped with the attention module achieves similar accuracy while requiring higher GFLOPs compared to the model equipped with SGS. The reason for such a great difference in GFLOPs is that attention modules perform a weighting of the features, while our approach clusters and reduces features. If all features are the same, the attention module should weight them equally while our approach reduces them to one feature.

6.5 Summary

Designing computationally efficient deep 3D convolutional neural networks for video understanding is a challenging task. In this chapter, we proposed a novel trainable module called Similarity Guided Sampling (SGS) to increase the efficiency of 3D CNNs for action recognition. The new SGS module selects the most informative and distinctive temporal features within a network such that as many temporal features as needed but not more than necessary are used for each input clip. By integrating SGS as an additional layer within current 3D CNNs, which use static temporal feature resolutions, we can convert them into much more efficient 3D CNNs with *adaptive temporal feature resolutions (ATFR)*. We evaluated our approach on six action recognition datasets and integrated SGS into five different state-of-the-art 3D CNNs. The results demonstrate that SGS drastically decreases the computation cost (GFLOPS) between 33% and 53% without compromising accuracy.

model	GFLOPs	top1
I3D* *Carreira and Zisserman* (2017)	108× N/A	71.1
I3D+SCSampler* *Korbar et al.* (2019)	108×10+N/A	75.1
Two-Stream I3D* *Carreira and Zisserman* (2017)	216×N/A	75.7
Two-Stream S3D-G* *Xie et al.* (2018b)	143×N/A	77.2
TSM R50* *Lin et al.* (2019)	65×10	74.4
Two-Stream I3D *Carreira and Zisserman* (2017)	216×N/A	75.7
R(2+1)D *Tran et al.* (2018b)	152×115	72.0
Two-Stream R(2+1)D *Tran et al.* (2018b)	304×115	73.9
FASTER32 *Zhu et al.* (2020)	67.7×8	75.3
SlowFast8×8,R101+NL *Feichtenhofer et al.* (2019)	116×30	78.7
SlowFast16×8,R101+NL *Feichtenhofer et al.* (2019)	234×30	79.8
X3D-L$^{\alpha}$ *Feichtenhofer* (2020)	18.3×10	76.8
X3D-L$^{\beta}$ *Feichtenhofer* (2020)	24.8×30	77.5
SlowFast4×16,R50 *Feichtenhofer et al.* (2019)	36.1×30	75.6
SlowFast4×16,R50**+ATFR**	20.8×30 ($\downarrow$ 42%)	75.8
X3D-S$^{\alpha}$ *Feichtenhofer* (2020)	1.9×10	72.9
X3D-S**+ATFR**$^{\alpha}$	**1.0** $\times$ 10 ($\downarrow$ 47%)	73.5
X3D-XL$^{\alpha}$ *Feichtenhofer* (2020)	35.8×10	78.4
X3D-XL**+ATFR**$^{\alpha}$	20×10 ($\downarrow$ 44%)	78.6
X3D-XL$^{\beta}$ *Feichtenhofer* (2020)	48.4×30	79.1
X3D-XL**+ATFR**$^{\beta}$	26.3×30 ($\downarrow$ 45%)	79.3
HATNET (chapter 5)	45.8×10	79.3
HATNET+ATFR	28.8×10($\downarrow$ 37%)	79.3
Swin-L (384) *Liu et al.* (2022)	2107×50	84.9
TokenLearner 16at18 (L/10) *Ryoo et al.* (2021)	4076×12	**85.4**

Table 6.8: Comparison to the state-of-the-art on Kinetics-400. Following *Feichtenhofer* (2020), we apply two testing strategies: $^{\alpha}$ samples uniformly 10 clips; $^{\beta}$ takes additionally 3 spatial crops for each sampled clip. For both setups, spatial scaling and cropping settings are as in the work by *Feichtenhofer* (2020). * denotes models pretrained on ImageNet. It is essential to highlight that the research presented in this chapter has been published in CVPR 2021 (*Fayyaz et al.*, 2021). To offer a comprehensive overview and enhanced understanding of the current trends in the field, we have incorporated recent state-of-the-art methods published after the publication date of this chapter.

model	pretrain	GFLOPs	top1
Oct-I3D+NL *Carreira and Zisserman* (2017)	ImageNet	25.6×30	76.0
I3D *Carreira and Zisserman* (2017)	-	108× N/A	71.9
SlowFast16×8,R101+NL *Feichtenhofer et al.* (2019)	-	234×30	81.8
SlowFast4×16,R50 *Feichtenhofer et al.* (2019)	-	36.1×30	78.8
HATNET (chapter 5)	-	45.8×10	81.6
HATNET+ATFR	-	28.8 × 10(↓ 37%)	81.6
X3D-M *Feichtenhofer* (2020)	-	6.2×30	78.8
X3D-M+**ATFR**	-	**3.3** × 30 (↓ 46%)	79.0
X3D-XL *Feichtenhofer* (2020)	-	48.4×30	81.9
X3D-XL+**ATFR**	-	25.6 × 30 (↓ 47%)	82.1
Swin-L (384) *Liu et al.* (2022)	ImageNet-21K	2107×50	**86.1**
TokenLearner 16at18 (L/10) *Ryoo et al.* (2021)	JFT	4076×12	**86.1**

Table 6.9: Comparison to the state-of-the-art on Kinetics-600. It is essential to highlight that the research presented in this chapter has been published in CVPR 2021 (*Fayyaz et al.*, 2021). To offer a comprehensive overview and enhanced understanding of the current trends in the field, we have incorporated recent state-of-the-art methods published after the publication date of this chapter.

model	pretrain	GFLOPs	top1	top5
SlowFast-R50 *Wu et al.* (2020)	Kinetics400	132.8	61.7	87.8
SlowFast-R50+**ATFR**	Kinetics400	**87.8** (↓ 33%)	**61.8**	**87.9**

Table 6.10: Results for the Something-Something-V2 dataset.

model	backbone	UCF101	HMDB51
C3D *Tran et al.* (2015)	RenNet18	89.8	62.1
RGB-I3D *Carreira and Zisserman* (2017)	Inception V1	95.6	74.8
R(2+1)D *Tran et al.* (2018b)	ResNet50	96.8	74.5
DynamoNet *Diba et al.* (2019)	ResNet101	96.6	74.9
HATNet *Diba et al.* (2020)	ResNet50	97.8	76.5
3DResNet+ATFR	ResNet50	97.9	76.7

Table 6.11: Comparison with other methods on UCF101 and HMDB51.

stage	layer	output size
raw	-	$32 \times 244 \times 224$
conv_1	$5 \times 7 \times 7, 8$, stride $1, 2, 2$	$32 \times 112 \times 112$
pool_1	$1 \times 3 \times 3$, max, stride $1, 2, 2$	$32 \times 56 \times 56$
res_2	$\begin{bmatrix} 3 \times 1 \times 1, 8 \\ 1 \times 3 \times 3, 8 \\ 1 \times 1 \times 1, 32 \end{bmatrix} \times 2$	$32 \times 56 \times 56$
res_3	$\begin{bmatrix} 1 \times 1 \times 1, 64 \\ 1 \times 3 \times 3, 64 \\ 1 \times 1 \times 1, 256 \end{bmatrix} \times 2$	$32 \times 28 \times 28$
res_4	$\begin{bmatrix} 1 \times 1 \times 1, 128 \\ 1 \times 3 \times 3, 128 \\ 1 \times 1 \times 1, 512 \end{bmatrix} \times 2$	$32 \times 14 \times 14$
res_5	$\begin{bmatrix} 3 \times 1 \times 1, 256 \\ 1 \times 3 \times 3, 256 \\ 1 \times 1 \times 1, 1024 \end{bmatrix} \times 2$	$32 \times 7 \times 7$
	global average pool, fc	$1 \times 1 \times 1$

Table 6.12: Modified 3DResNet-18

Model	Train	Inference (fps)
X3D-S	131h	2834
X3D-S+ATFR	**121h**	**4295**

Table 6.13: Runtime on Kinetics-400.

B	4	8	16	32
top1	61.4	64.7	64.7	69.6
top5	86.3	85.8	86.2	88.8
GFLOPs	3.5	4.2	5.5	14.0

Table 6.14: Ablations on the effect of changing the numbers of bins B for 3DResNet-18+ATFR on Mini-Kinetics. The model is trained and validated for different number of bins. We show top-1 and top-5 classification accuracy (%).

B	4	8	16	32
top1	51.1	61.4	64.4	69.6
top5	75.1	83.5	85.5	88.8
GFLOPs	3.5	5.0	8.0	14.0

Table 6.15: Ablations on the effect of changing the numbers of bins B only during inference for 3DResNet-18+ATFR on Mini-Kinetics. The model is trained with 32 bins, but inference is performed with a different number of bins. We show top-1 and top-5 classification accuracy (%).

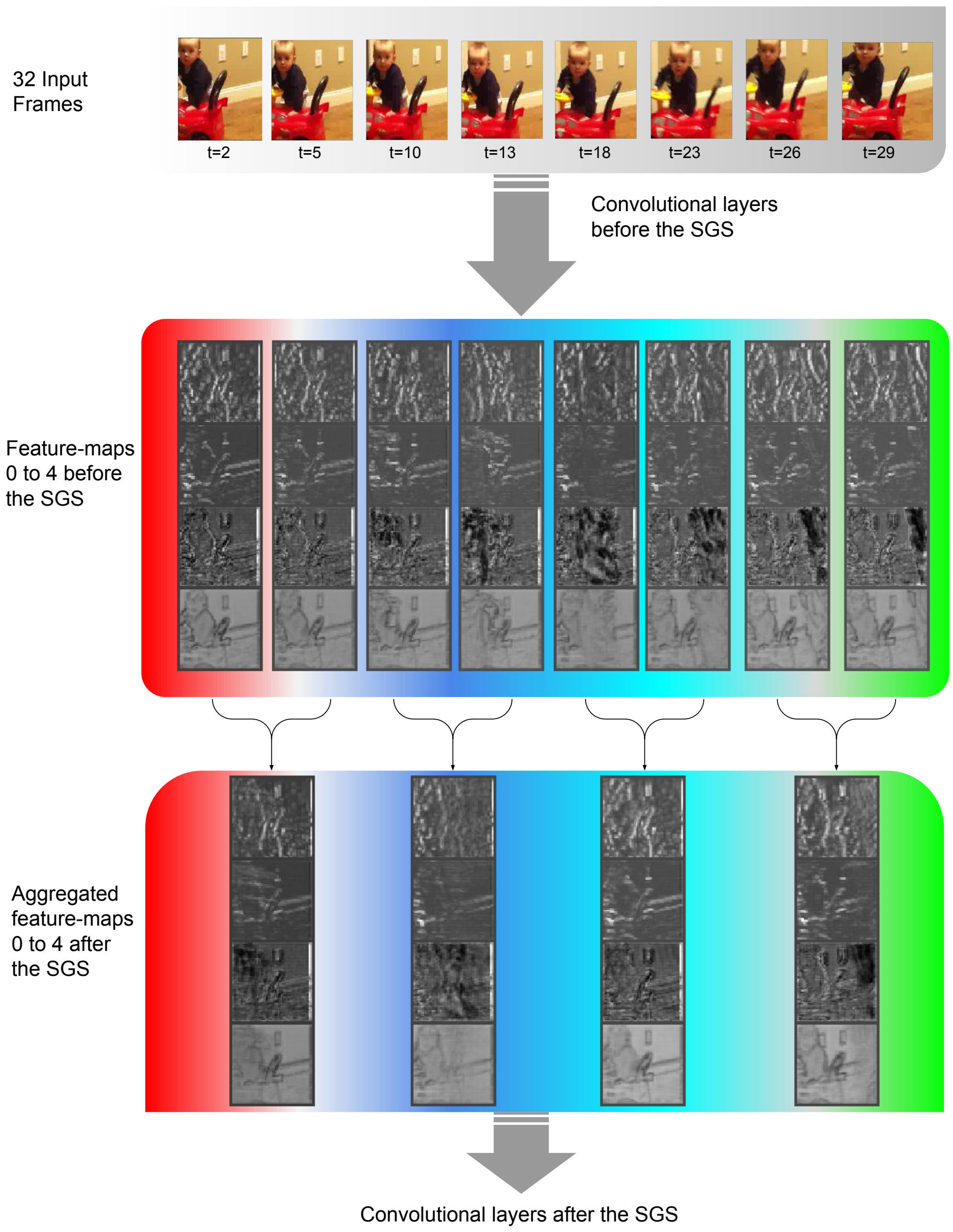

32 Input Frames
t=2
t=5
t=10
t=13
t=18
t=23
t=26
t=29
Convolutional layers before the SGS
Feature-maps 0 to 4 before the SGS
Aggregated feature-maps 0 to 4 after the SGS
Convolutional layers after the SGS

Figure 6.5: Visualization of the feature maps of 3DResNet-50+ATFR with linear kernel. In the first row, 8 frames out of 32 input frames are shown. The corresponding temporal feature maps of *ResBlock 2* are depicted in the second row. The third row shows the aggregated feature maps after the SGS. Note that we only show the first 4 channels of the feature maps for better visualization.

Model	top1	GFLOPs
X3D-S	77.9	1.9
X3D-S+ATFR	78.0	**1.1**
X3D-S+Temporal Attention	78.3	1.9

Table 6.16: Comparison with attention modules. The models are trained and tested on the Mini-Kinetics dataset.

CHAPTER 7

Adaptive Vision Transformers for Efficient Video Classification

This chapter is based on the following publication:
Adaptive Token Sampling for Efficient Vision Transformers
Mohsen Fayyaz*, Soroush Abbasi Kouhpayegani*, Farnoush Rezaei Jafari*, Sunando Sengupta, Hamid Reza Vaezi Joze, Eric Sommerlade, Hamed Pirsiavash, Juergen Gall
European Conference on Pattern Recognition (ECCV), 2022.
* *denotes equal contribution.*

Below we will summarize the contributions of each author.

- **Mohsen Fayyaz, Soroush Abbasi Kouhpayegani, and Farnoush Rezaei Jafari**
  This work represents a collaborative effort between Mohsen Fayyaz, Soroush Abbasi Kouhpayegani, and Farnoush Rezaei Jafari, with each author contributing significantly to the conceptualization, implementation, and empirical evaluation of the proposed adaptive token sampling (ATS) module for vision transformers. Mohsen Fayyaz initially conceptualized the ATS module and the idea of vision transformers with adaptive token resolutions, while Soroush Abbasi Kouhpayegani proposed and evaluated various sampling approaches and provided the final formulation of the inverse transform sampling used in the ATS module. Farnoush Rezaei Jafari evaluated the concept of token sampling through the use of explainable artificial intelligence (XAI) methods and implemented the inverse transform sampling approach that ultimately achieved the best performance. All authors contributed to the preparation of the manuscript, with each playing a crucial role in the development of the proposed approach. It is worth noting that the authors contributed equally to this work.

- **Sunando Sengupta and Hamid Reza Vaezi Joze**
  The authors made substantial contributions to this work through their discussions and writing.

- **Eric Sommerlade, Hamed Pirsiavash, and Juergen Gall**
  This work was made possible through the invaluable supervision and guidance provided by Eric Sommerlade, Hamed Pirsiavash, and Juergen Gall, as well as their contributions through suggestions, discussions, and writing. Additionally, Eric Sommerlade funded and managed the project.

Contents

In the previous chapter, we demonstrated the effectiveness of adaptively adjusting temporal feature resolutions in 3D Convolutional Neural Networks (CNNs) for improved efficiency. In this chapter, we present a method for similarly optimizing the efficiency of state-of-the-art vision transformers through the use of an Adaptive Token Sampler (ATS) module. Vision transformer models have achieved impressive results on a range of vision tasks, including image and video classification, but their high computational demands and GFLOP requirements can be limiting. While the GFLOPs of a vision transformer can be reduced by decreasing the number of tokens within the network, there is no optimal setting that applies to all input images or videos. The proposed ATS module addresses this issue by scoring and adaptively sampling significant tokens, resulting in a variable number of tokens that is optimized for each individual input image. By integrating the ATS module into transformer blocks, we are able to significantly improve the efficiency of vision transformers without sacrificing accuracy. Importantly, the ATS module is parameter-free and can be easily incorporated into pre-trained vision transformers as a plug-and-play module, enabling the reduction of GFLOPs without the need for additional training. Additionally, the differentiable design of the ATS module allows for the training of a vision transformer equipped with the module. In our evaluations on the ImageNet, Kinetics-400, and Kinetics-600 datasets, we demonstrate the effectiveness of the ATS module in both image and video classification tasks by integrating it into multiple state-of-the-art vision transformers, achieving a reduction in GFLOPs of approximately a factor of 2 while preserving strong performance.

7.1 Introduction

Over the last ten years, there has been a tremendous progress on image and video understanding in the light of new and complex deep learning architectures, which are based on the variants of 2D (*He et al.*, 2016a; *Simonyan and Zisserman*, 2015; *Krizhevsky et al.*, 2012) and 3D (*Tran et al.*, 2015; *Feichtenhofer et al.*, 2019; *Diba et al.*, 2018, 2019; *Tran et al.*, 2017; *Feichtenhofer*, 2020) Convolutional Neural Networks (CNNs). Recently, vision transformers have shown promising results in image classification (*Dosovitskiy et al.*, 2021; *Touvron et al.*, 2021; *Jiang et al.*, 2021; *Wu et al.*, 2021) and action recognition (*Bertasius et al.*, 2021; *Liu et al.*, 2021; *Bulat et al.*, 2021) compared to CNNs. Although vision transformers have a superior representation power, the high computational cost of their transformer blocks make them unsuitable for many edge devices. The computational cost of a vision transformer grows quadratically with respect to the number of tokens it uses. To reduce the number of tokens and thus the computational cost of a vision transformer, DynamicViT (*Rao et al.*, 2021a) proposes a token scoring neural network to predict which tokens are redundant.

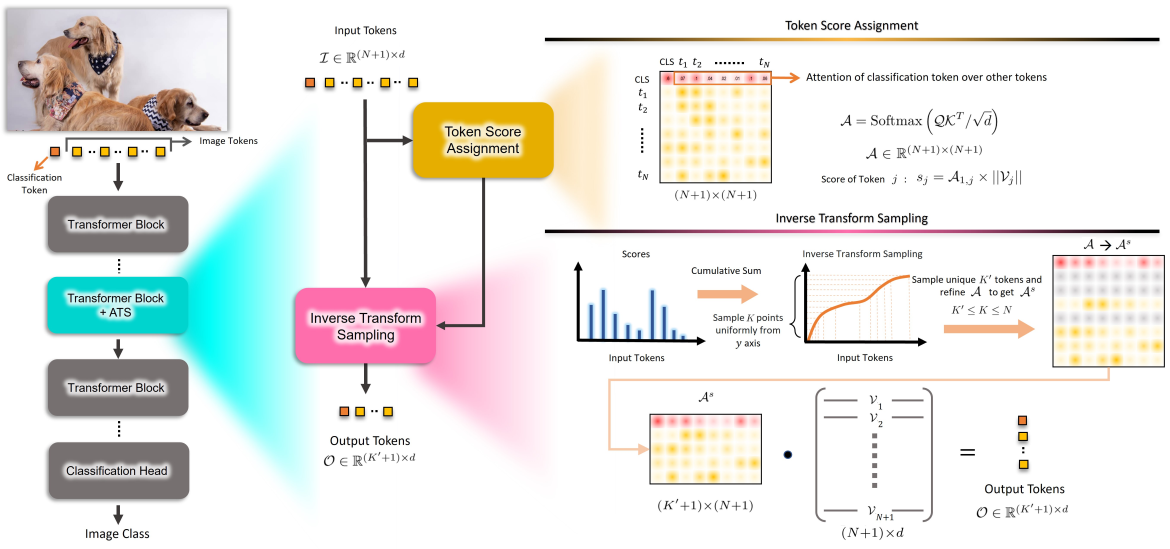


Image Tokens
Classification Token
Transformer Block
Transformer Block + ATS
Transformer Block
Classification Head
Image Class
Input Tokens
$\mathcal{I} \in \mathbb{R}^{(N+1)\times d}$
Token Score Assignment
Inverse Transform Sampling
Output Tokens
$\mathcal{O} \in \mathbb{R}^{(K'+1)\times d}$
Token Score Assignment
Attention of classification token over other tokens
$\mathcal{A} = \text{Softmax}\left(\mathcal{QK}^T/\sqrt{d}\right)$
$\mathcal{A} \in \mathbb{R}^{(N+1)\times(N+1)}$
Score of Token j : $s_j = \mathcal{A}_{1,j} \times ||\mathcal{V}_j||$
$(N+1)\times(N+1)$
Inverse Transform Sampling
Scores
Cumulative Sum
Sample K points uniformly from y axis
Inverse Transform Sampling
Sample unique K' tokens and refine $\mathcal{A}$ to get $\mathcal{A}^s$
$K' \leq K \leq N$
$\mathcal{A} \rightarrow \mathcal{A}^s$
Input Tokens
Input Tokens
$\mathcal{A}^s$
$(K'+1)\times(N+1)$
$(N+1)\times d$
Output Tokens
$\mathcal{O} \in \mathbb{R}^{(K'+1)\times d}$

Figure 7.1: The Adaptive Token Sampler (ATS) can be integrated into the self-attention layer of any transformer block of a vision transformer model (left). The ATS module takes at each stage a set of input tokens $\mathcal{I}$. The first token is considered as the classification token in each block of the vision transformer. The attention matrix $\mathcal{A}$ is then calculated by the dot product of the queries $\mathcal{Q}$ and keys $\mathcal{K}$, scaled by $\sqrt{d}$. We use the attention weights $\mathcal{A}_{1,2}, \ldots, \mathcal{A}_{1,N+1}$ of the classification token as significance scores $\mathcal{S} \in \mathbb{R}^N$ for pruning the attention matrix $\mathcal{A}$. To reflect the effect of values $\mathcal{V}$ on the output tokens $\mathcal{O}$, we multiply the $\mathcal{A}_{1,j}$ by the magnitude of the corresponding value $\mathcal{V}_j$. We select the significant tokens using inverse transform sampling over the cumulative distribution function of the scores $\mathcal{S}$. Having selected the significant tokens, we then sample the corresponding attention weights (rows of the attention matrix $\mathcal{A}$) to get $\mathcal{A}^s$. Finally, we softly downsample the input tokens $\mathcal{I}$ to output tokens $\mathcal{O}$ using the dot product of $\mathcal{A}^s$ and $\mathcal{V}$.

The approach then keeps a fixed ratio of tokens at each stage. Although DynamicViT reduces the GFLOPs of a given network, its scoring network introduces an additional computational overhead. Furthermore, the scoring network needs to be trained together with the vision transformer and it requires to modify the loss function by adding additional loss terms and hyper-parameters. To alleviate such limitations, EViT (*Liang et al.*, 2022) employs the attention weights as the tokens' importance scores. A further limitation of both EViT and DynamicViT is that they need to be re-trained if the fixed target ratios need to be changed (*e.g.* due to deployment on a different device). This strongly limits their applications.

In this work, we propose a method to efficiently reduce the number of tokens in any given vision transformer without the mentioned limitations. Our approach is motivated by the observation that in image/action classification, all parts of an input image/video do not contribute equally to the final classification scores and some parts contain irrelevant or redundant information. The amount of relevant information varies depending on the content of an image or video. For instance, in Figure 7.10, we can observe examples in which only a few or many patches are required for correct classification. The same holds for the number of tokens used at each stage, as illustrated in Figure 7.2. Therefore, we propose an approach that automatically selects an adequate number of tokens at each stage based on the image content, *i.e.* the number of the selected tokens at all network's stages varies for different images, as shown in Figure 7.9. It is in contrast to the works by *Rao et al.* (2021a) and *Liang et al.* (2022), where the ratio of the selected tokens needs to be specified for each stage and is constant after training. However, selecting a static number of tokens will on the one hand discard important information for challenging images/videos, which leads to a classification accuracy drop. On the other hand, it will use more tokens than necessary for the easy cases and thus waste computational resources. In this work, we address the question of how a transformer can dynamically adapt its computational resources in a way that not more resources than necessary are used for each input image/video.

To this end, we introduce a novel *Adaptive Token Sampler (ATS)* module. ATS is a differentiable parameter-free module that adaptively down-samples input tokens. To do so, we first assign significance scores to the input tokens by employing the attention weights of the classification token in the self-attention layer and then select a subset of tokens using inverse transform sampling over the scores. Finally, we softly down-sample the output tokens to remove redundant information with the least amount of information loss. In contrast to *Rao et al.* (2021a), our approach does not add any additional learnable parameters to the network. While the ATS module can be added to any off-the-shelf pre-trained vision transformer without any further training, the network equipped with the differentiable ATS module can also be further fine-tuned. Moreover, one may train a model only once and then adjust a maximum limit for the ATS module to adapt it to the resources of different edge devices at the inference time. This eliminates the need of training separate models for different levels of computational resources.

We demonstrate the efficiency of our proposed adaptive token sampler for image classification by integrating it into the current state-of-the-art vision transformers such as DeiT (*Touvron et al.*, 2021), CvT (*Wu et al.*, 2021), and PS-ViT (*Yue et al.*, 2021). As shown in Figure 7.5, our approach significantly reduces the GFLOPs of vision transformers of various sizes without significant loss of accuracy. We evaluate the effectiveness of our method by comparing it with other methods designed for reducing the number of tokens, including DynamicViT (*Rao et al.*, 2021a), EViT (*Liang et al.*, 2022), and Hierarchical Pooling (*Pan et al.*, 2021b). Extensive experiments on the ImageNet dataset show that our method outperforms existing approaches and provides the best trade-off between computational cost and classification accuracy. We also demonstrate the efficiency of our proposed module for action recognition by adding it to the state-of-the-art video vision transformers such as XViT (*Bulat et al.*, 2021) and TimeSformer (*Bertasius et al.*, 2021). Extensive experiments on the Kinetics-400 and Kinetics-600 datasets show that our method surpasses the performance of existing approaches and leads to the best computational cost/accuracy trade-off. In a nutshell, the adaptive token sampler can significantly scale down the off-the-shelf vision transformers' computational costs and it is therefore very useful for real-world vision-based applications.

7.2 Adaptive Token Sampler

State-of-the-art vision transformers are computationally expensive since their computational costs grow quadratically with respect to the number of tokens, which is static at all stages of the network and corresponds to the number of input patches. Convolutional neural networks deal with the computational cost by reducing the resolution within the network using various pooling operations. It means that the spatial or temporal resolution decreases at the later stages of the network. However, applying such simple strategies, *i.e.* pooling operations with fixed kernels, to vision transformers is not straightforward since the tokens are permutation invariant. Moreover, such static down-sampling approaches are not optimal. On the one hand, a fixed down-sampling method discards important information at some locations of the image or video, like details of the object. On the other hand, it still includes many redundant features that do not contribute to the classification accuracy, for instance, when dealing with an image with a homogeneous background. Therefore, we propose an approach that dynamically adapts the number of tokens at each stage of the network based on the input data such that important information is not discarded and no computational resources are wasted for processing redundant information.

To this end, we propose our novel Adaptive Token Sampler (ATS) module. ATS is a parameter-free differentiable module to sample significant tokens over the input tokens. In our ATS module, we first assign significance scores to the N input tokens and then select a subset of these tokens based on their scores. The upper bound of GFLOPs can be set by defining a maximum limit for the number of tokens sampled, denoted by K. Since the sampling procedure can sample some input tokens several times, we only keep one instance of a token. The number of sampled tokens K' is thus usually lower than K and varies among input images or videos (Figure 7.9). Figure 7.1 gives an overview of our proposed approach.

7.2.1 Token Scoring

Let $\mathcal{I} \in \mathbb{R}^{(N+1)\times d}$ be the input tokens of a self-attention layer with $N+1$ tokens. Before forwarding the input tokens through the model, ViT concatenates a classification token to the input tokens. The corresponding output token at the final transformer block is then fed to the classification head to get the class probabilities. Practically, this token is placed as the first token in each block and it is considered as a classification token. While we keep the classification token, our goal is to reduce the output tokens $\mathcal{O} \in \mathbb{R}^{(K'+1)\times d}$ such that K' is dynamically adapted based on the input image or video and $K' \leqslant K \leqslant N$, where K is a parameter that controls the maximum number of sampled tokens. Figure 7.9 shows how the number of sampled tokens K' varies for different input data and stages of a network. We first describe how each token is scored.

In a standard self-attention layer (Vaswani *et al.*, 2017), the queries $\mathcal{Q} \in \mathbb{R}^{(N+1)\times d}$, keys $\mathcal{K} \in \mathbb{R}^{(N+1)\times d}$, and values $\mathcal{V} \in \mathbb{R}^{(N+1)\times d}$ are computed from the input tokens $\mathcal{I} \in \mathbb{R}^{(N+1)\times d}$. The attention matrix $\mathcal{A}$ is then calculated by the dot product of the queries and keys, scaled by $\sqrt{d}$:

$$\mathcal{A} = \left(\mathcal{Q}\mathcal{K}^T/\sqrt{d}\right). \tag{7.1}$$

Due to the function, each row of $\mathcal{A} \in \mathbb{R}^{(N+1)\times(N+1)}$ sums up to 1. The output tokens are then calculated using a combination of the values weighted by the attention weights:

$$\mathcal{O} = \mathcal{A}\mathcal{V}. \tag{7.2}$$

Each row of $\mathcal{A}$ contains the attention weights of an output token. The weights indicate the contributions of all input tokens to the output token. Since $\mathcal{A}_{1,:}$ contains the attention weights of the classification token, $\mathcal{A}_{1,j}$ represents the importance of the input token j for the output classification token. Thus, we use the weights $\mathcal{A}_{1,2}, \ldots, \mathcal{A}_{1,N+1}$ as significance scores for pruning the attention matrix $\mathcal{A}$, as illustrated in Figure 7.1. Note that $\mathcal{A}_{1,1}$ is not used since we keep the classification token. As the output tokens $\mathcal{O}$ depend on both $\mathcal{A}$ and $\mathcal{V}$ (7.2), we also take into account the norm of $\mathcal{V}_j$ for calculating the j^{th} token's significance score. The motivation is that values having a norm close to zero have a low impact and their corresponding tokens are thus less significant. In our experiments, we show that multiplying $\mathcal{A}_{1,j}$ with the norm of $\mathcal{V}_j$ improves the results. The significance score of a token j is thus given by

$$\mathcal{S}_j = \frac{\mathcal{A}_{1,j} \times ||\mathcal{V}_j||}{\sum_{i=2} \mathcal{A}_{1,i} \times ||\mathcal{V}_i||} \tag{7.3}$$

where $i, j \in \{2 \ldots N\}$. For a multi-head attention layer, we calculate the scores for each head and then sum the scores over all heads.

7.2.2 Token Sampling

Having computed the significance scores of all tokens, we can prune their corresponding rows from the attention matrix $\mathcal{A}$. To do so, a naive approach is to select K tokens with the highest significance scores (top-K selection). However, this approach does not perform well, as we show in our experiments and it can not adaptively select $K' \leqslant K$ tokens. is that it discards all tokens with lower scores. Some of these tokens, however, can be useful in particular at the earlier stages when the features are less discriminative. For instance, having multiple tokens with similar keys, which may occur in the early stages, will lower their corresponding attention weights due to the function. Although one of these tokens would be beneficial at the later stages, taking the top-K tokens might discard all of them. Therefore, we suggest sampling tokens based on their significance scores. In this case, the probability of sampling one of the several similar tokens is equal to the sum of their scores. We also observe that the proposed sampling procedure selects more tokens at the earlier stages than the later stages as shown in Figure 7.2.

For the sampling step, we suggest using inverse transform sampling to sample tokens based on their significance scores $\mathcal{S}$ (7.3). Since the scores are normalized, they can be interpreted as probabilities and we can calculate the cumulative distribution function () of $\mathcal{S}$:

$${}_i = \sum_{j=2}^{j=i} \mathcal{S}_j. \tag{7.4}$$

Note that we start with the second token since we keep the first token. Having the cumulative distribution function, we obtain the sampling function by taking the inverse of the :

$$\Psi(k) =^{-1} (k) \tag{7.5}$$

where $k \in [0, 1]$. In other words, the significance scores are used to calculate the mapping function between the indices of the original tokens and the sampled tokens. To obtain K samples, we can sample K-times from the uniform distribution $U[0, 1]$. While such randomization might be desirable for some applications, deterministic inference is in most cases preferred. Therefore, we use a fixed sampling scheme for training and inference by choosing $k = \{\frac{1}{2K}, \frac{3}{2K} \ldots, \frac{2K-1}{2K}\}$. Since $\Psi(.) \in \mathbb{R}$, we consider the indices of the tokens with the nearest significant scores as the sampling indices.

If a token is sampled more than once, we only keep one instance. As a consequence, the number of unique indices K' is often lower than K as shown in Figure 7.9. In fact, $K' < K$ if there is at least one token with a score $S_j \geqslant 2/K$. In the two extreme cases, either only one dominant token is selected and $K' = 1$ or $K' = K$ if the scores are more or less balanced. Interestingly, more tokens are selected at the earlier stages, where the features are less discriminative and the attention weights are more balanced, and less at the later stages, as shown in Figure 7.2. The number and locations of tokens also vary for different input images, as shown in Figure 7.10. For images with a homogeneous background that covers a large part of the image, only a few tokens are sampled. In this case, the tokens cover the object in the foreground and are sparsely but uniformly sampled from the background. In cluttered images, many tokens are required. It illustrates the importance of making the token sampling procedure adaptive.

Having indices of the sampled tokens, we refine the attention matrix $\mathcal{A} \in \mathbb{R}^{(N+1)\times(N+1)}$ by selecting the rows that correspond to the sampled $K'+1$ tokens. We denote the refined attention matrix by $\mathcal{A}^s \in \mathbb{R}^{(K'+1)\times(N+1)}$. To obtain the output tokens $\mathcal{O} \in \mathbb{R}^{(K'+1)\times d}$, we thus replace the attention matrix $\mathcal{A}$ by the refined one $\mathcal{A}^s$ in (7.2) such that:

$$\mathcal{O} = \mathcal{A}^s \mathcal{V}. \tag{7.6}$$

These output tokens are then taken as input for the next stage. In our experimental evaluation, we demonstrate the efficiency of the proposed adaptive token sampler, which can be added to any vision transformer.

7.3 Experiments

In this section, we analyze the performance of our ATS module by adding it to different backbone models and evaluating them on ImageNet (*Deng et al.*, 2009), Kinetics-400 (*Kay et al.*, 2017), and Kinetics-600 (*Carreira et al.*, 2018), which are large-scale image and video classification datasets, respectively. In addition, we perform several ablation studies to better analyze our method. For the image classification task, we evaluate our proposed method on the ImageNet (*Deng et al.*, 2009) dataset with 1.3M images and 1K classes. For the action classification task, we evaluate our approach on the Kinetics-400 (*Kay et al.*, 2017) and Kinetics-600 (*Carreira et al.*, 2018) datasets with 400 and 600 human action classes, respectively. We use the standard training/testing splits and protocols provided by the ImageNet and Kinetics datasets. If not otherwise stated, the number of output tokens of the ATS module are limited by the number of its input tokens. For example, we set $K = 197$ in case of DeiT-S (*Touvron et al.*, 2021). For the image classification task, we follow the fine-tuning setup of (*Rao et al.*, 2021a) if not mentioned otherwise. The fine-tuned models are initialized by their backbones' pre-trained weights and trained for 30 epochs using PyTorch AdamW optimizer (lr= $5{-}4$, batch size = 8×96). We use the cosine scheduler for training the networks. For more implementation details and also information regarding action classification models, please refer to the supplementary materials.

7.3.1 Ablation Experiments

First, we analyze different setups for our ATS module. Then, we investigate the efficiency and effects of our ATS module when incorporated in different models. If not otherwise stated, we use the pre-

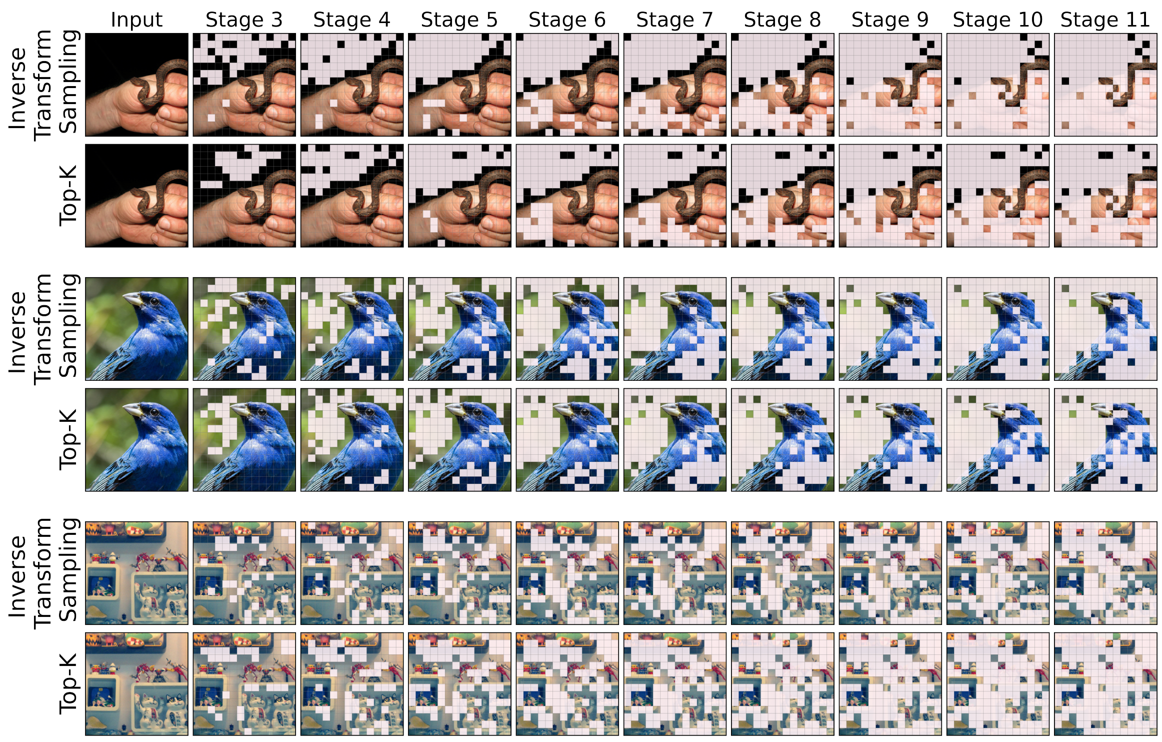


Figure 7.2: Visualization of the gradual token sampling procedure in the multi-stage DeiT-S+ATS model. As it can be seen, at each stage, those tokens that are considered to be less significant to the classification are masked and the ones that have contributed the most to the model's prediction are sampled. We also visualize the token sampling results with Top-K selection to have a better comparison to our Inverse Transform Sampling.

trained DeiT-S (*Touvron et al.*, 2021) model as the backbone and we do not fine-tune the model after adding the ATS module. We integrate the ATS module into stage 3 of the DeiT-S (*Touvron et al.*, 2021) model. We report the results on the ImageNet-1K validation set in all of our ablation studies.

7.3.1.1 Significance Scores

As mentioned in Sec. 7.2.1, we use the attention weights of the classification token as significance scores for selecting our candidate tokens. In this experiment, we evaluate different approaches for calculating significance scores. Instead of directly using the attention weights of the classification token, we sum over the attention weights of all tokens (rows of the attention matrix) to find tokens with highest significance over other tokens. We show the results of this method in Figure 7.4 labeled as Self-Attention score. As it can be seen, using the attention weights of the classification token performs better specially in lower FLOPs regimes. The results show that the attention weights of the classification token are a much stronger signal for selecting the candidate tokens. The reason for this is that the classification token will later be used to predict the class probabilities in the final stage of the model. Thus, its corresponding attention weights show which tokens have more impact on the output classification token. Whereas summing over all attention weights only shows us the tokens

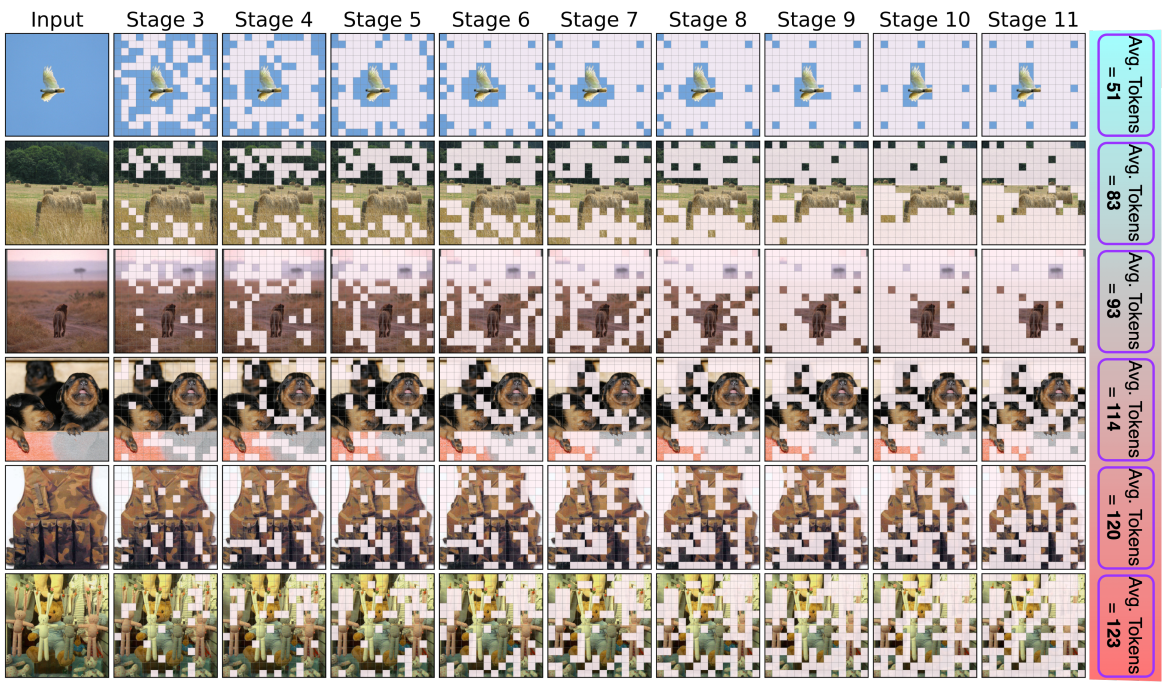


Figure 7.3: Visualization of the gradual token sampling procedure in the multi-stage DeiT-S+ATS model. We integrate our ATS module into the stages 3 to 11 of the DeiT-S model. The tokens that are sampled at each stage of the network are shown for images that are ordered by their complexity (from low complexity to high complexity). We visualize the tokens, which are discarded, as masks over the input images. As it can be seen, a higher number of tokens are sampled for more cluttered images while a lower number of tokens are required when the images contain less details. Additionally, we can see that the sampled tokens are more focused and less scattered in images with less details.

with highest attention from all other tokens, which may not necessarily be useful for the classification token. To better investigate this observation, we also randomly select another token rather than the classification token and use its attention weights for the score assignment. As shown, this approach performs much worse than the other ones both in high and low FLOPs regimes. We also investigate the impact of using the L_2 norm of the values in (7.3). As it can be seen in Figure 7.4, it improves the results by about 0.2%.

In all of the above mentioned experiments, we suggested keeping the classification token since the loss is defined on this token and discarding it may negatively affect the performance. To represent the importance of this token experimentally, we sum over the attention weights of all tokens (rows of the attention matrix) to find the most significant tokens. We show this in Figure 7.11 as Self-Attention Score (CLS Enforced). In contrast to our previous experiments, we allow ATS to remove the classification token when it is of low importance based on the significance scores S. We show the results of this experiment in Figure 7.11 as Self-Attention Score (CLS Not Enforced). As it can be seen in Figure 7.11, discarding the classification token reduces the top-1 accuracy.

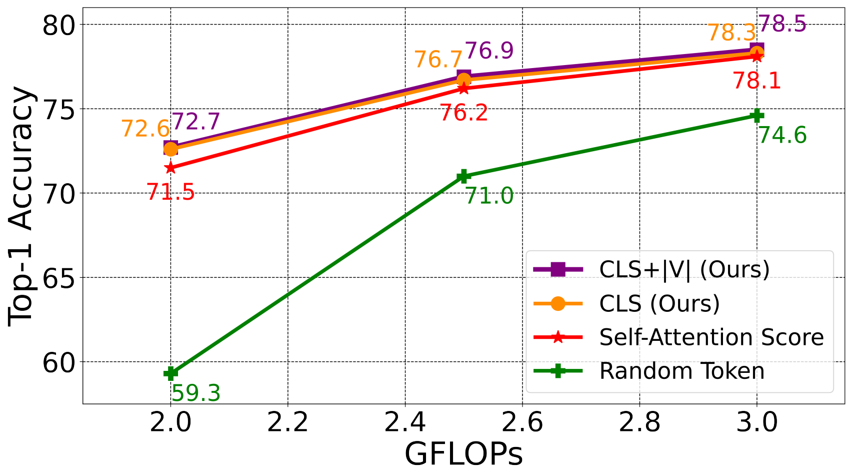


Figure 7.4: Impact of different score assignment methods. To achieve different GFLOPs levels, we bound the value of K from above such that the average GFLOPs of our adaptive models over the ImageNet validation set reaches the desired level.

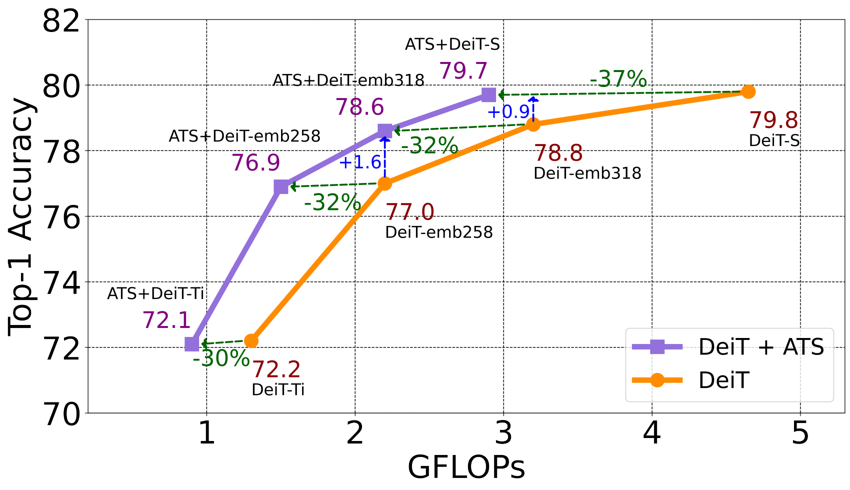


Figure 7.5: Performance comparison on the ImageNet validation set. Our proposed adaptive token sampling method achieves a state-of-the-art trade-off between accuracy and GFLOPs. We can reduce the GFLOPs of DeiT-S by 37% while almost maintaining the accuracy.

7.3.1.2 Candidate Tokens Selection

As mentioned in Sec. 7.2.2, we employ the inverse transform sampling approach to softly downsample the input tokens. To better investigate this approach, we also evaluate the model's performance when picking the top K tokens with highest significance scores $\mathcal{S}$. As it can be seen in Figure 7.6, our inverse transform sampling approach outperforms the Top-K selection both in high and low GFLOPs regimes. As discussed earlier, our inverse transform sampling approach based on the CDF of the scores does not hardly discard all tokens with lower significance scores and hence provides a more

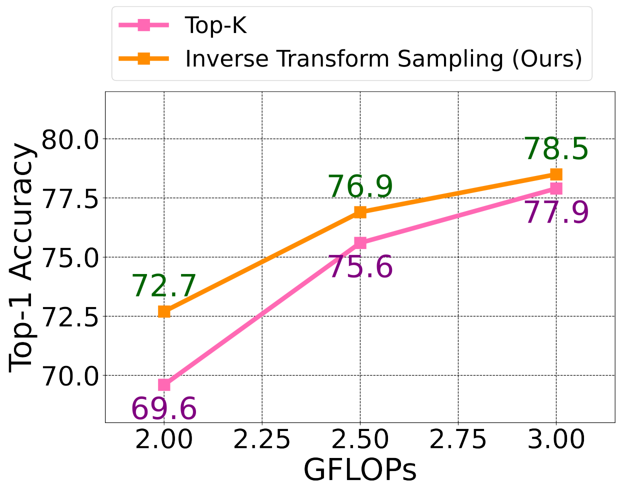


Figure 7.6: Impact of token sampling approaches. For the model with Top-K selection (fixed-rate sampling), we set K such that the model operates at a desired GFLOPs level. We control the GFLOPs level of our adaptive models as in Figure 7.4. We use DeiT-S (*Touvron et al.*, 2021) for these experiments.

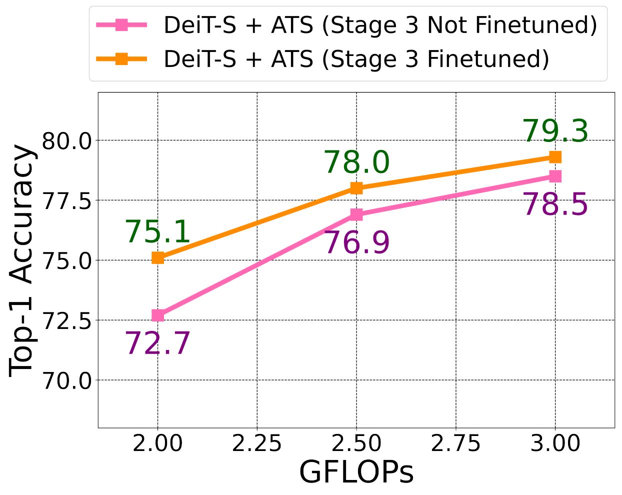


Figure 7.7: Impact of finetuning on ATS+Deit-S. We control the GFLOPs level of our adaptive models as in Figure 7.4. We use DeiT-S (*Touvron et al.*, 2021) for these experiments.

diverse set of tokens for the following layers. Since earlier transformer blocks are more prone to predict noisier attention weights for the classification token, such a diversified set of tokens can better contribute to the output classification token of the final transformer block. Moreover, the Top-K selection method will result in a fixed token selection rate at every stage that limits the performance of the backbone model. This is shown by the examples in Figure 7.2. For a cluttered image (bottom), inverse transform sampling keeps a higher number of tokens across all transformer blocks compared to the Top-K selection and hence preserves the accuracy. On the other hand, for a less detailed image (top), inverse transform sampling will retain less tokens, which results in less computation cost.

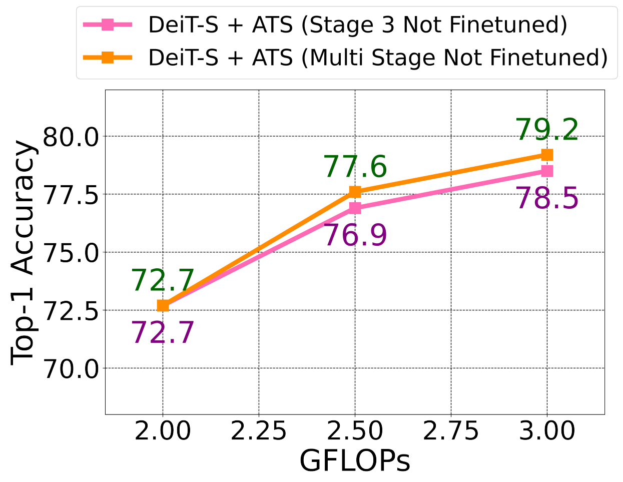


Figure 7.8: Impact of using ATS in multiple stages. We control the GFLOPs level of our adaptive models as in Figure 7.4. We use DeiT-S (*Touvron et al.*, 2021) for these experiments.

We also evaluate the performance of our trained multi-stage DeiT-S+ATS model when picking the top K tokens with the highest significance scores $\mathcal{S}$. To this end, we trained our DeiT-S+ATS network with the top-K selection approach and compared it to our DeiT-S+ATS model with the inverse transform sampling method. As it can be seen in Table 7.1, our inverse transform sampling approach outperforms the top-K selection with and without training (Figure 7.6). As discussed earlier, our inverse transform sampling approach does not hardly discard all tokens with lower significance scores and hence provides a more diverse set of tokens for the following layers. This sampling strategy also helps the model to gain a better performance after training, thanks to a more diversified token selection.

7.3.1.3 Model Scaling

Another common approach for changing the GFLOPs/accuracy trade-off of networks is to change the channel dimension. To demonstrate the efficiency of our adaptive token sampling method, we thus vary the dimensionality. To this end, we first train several DeiT models with different embedding dimensions. Then, we integrate our ATS module into the stages 3 to 11 of these DeiT backbones and fine-tune the networks. In Figure 7.5, we can observe that our approach can reduce GFLOPs by 37% while maintaining the DeiT-S backbone's accuracy. We can also observe that the GFLOPs reduction rate gets higher as we increase the embedding dimensions from 192 (DeiT-Ti) to 384 (DeiT-S). The results show that our ATS module can reduce the computation cost of the models with larger embedding dimensions to their variants with smaller embedding dimensions.

7.3.1.4 Visualizations

To better understand the way our ATS module operates, we visualize our token sampling procedure (Inverse Transform Sampling) in Figure 7.2. We have incorporated our ATS module in the stages 3 to 11 of the DeiT-S network. The tokens that are discarded at each stage are represented as a mask over the input image. We observe that our DeiT-S+ATS model has gradually removed irrelevant tokens

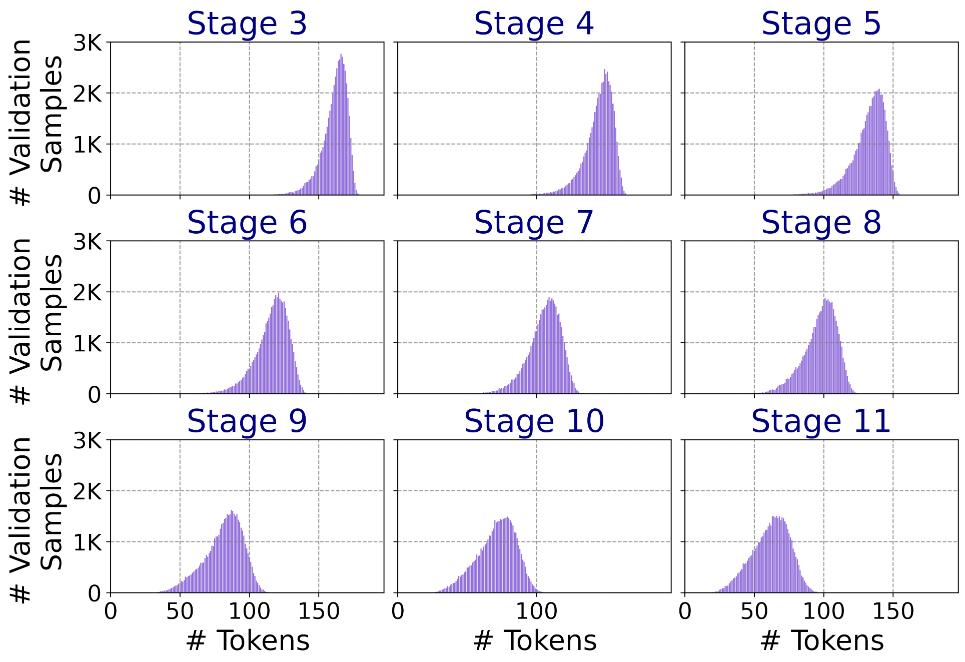


Figure 7.9: Histogram of the number of sampled tokens at each ATS stage of our multi-stage DeiT-S+ATS model on the ImageNet validation set. The y-axis corresponds to the number of images and the x-axis to the number of sampled tokens.

and sampled those tokens which are more significant to the model's prediction. In both examples, our method identified the tokens that are related to the target objects as the most informative tokens. We show more visual results in Figure 7.3. We select several images of the ImageNet validation set with various amounts of detail and complexity. We visualize the progressive token sampling procedure of our multi-stage DeiT-S+ATS model for the selected images. The number of output tokens of each ATS module in the multi-stage DeiT-S+ATS model is limited by the number of its input tokens, which is 197. Our adaptive model samples a higher number of tokens when the input images are more cluttered. We can also observe that the sampled tokens are more scattered in images with more details compared to more plain images.

7.3.1.5 Adaptive Sampling

In this experiment, we investigate the adaptivity of our token sampling approach. We evaluate our multi-stage DeiT-S+ATS model on the ImageNet validation set. In Figure 7.9, we visualize histograms of the number of sampled tokens at each ATS stage. We can observe that the number of selected tokens varies at all stages and for all images. We also qualitatively analyze this nice property of our ATS module in Figs. 7.2 and 7.10. We can observe that our ATS module selects a higher number of tokens when it deals with detailed and complex images while it selects a lower number of tokens for less detailed images.

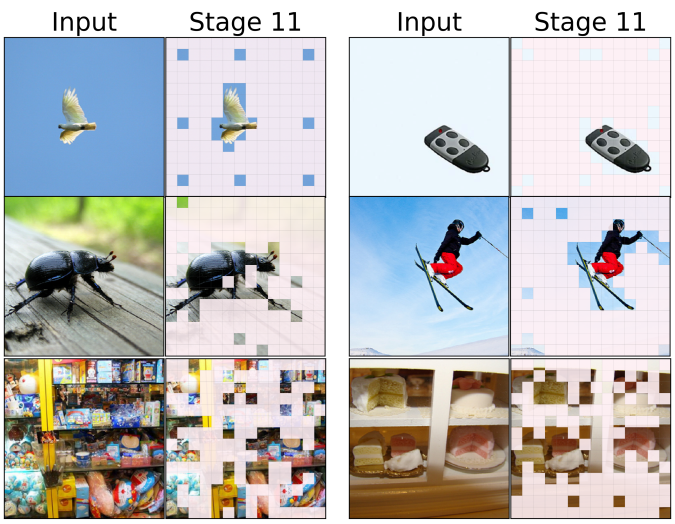


Figure 7.10: ATS samples less tokens for images with fewer details (top), and a higher number of tokens for more detailed images (bottom). We show the token downsampling results after all ATS stages. For this experiment, we use a multi-stage Deit-S+ATS model.

7.3.1.6 Fine-tuning

To explore the influence of fine-tuning on the performance of our approach, we fine-tune a DeiT-S+ATS model on the ImageNet training set. We compare the model with and without fine-tuning. As shown in Figure 7.7, fine-tuning can improve the accuracy of the model. In this experiment, we fine-tune the model with $K = 197$ but test it with different K values to reach the desired GFLOPs levels.

7.3.1.7 ATS Stages

In this experiment, we explore the effect of single-stage and multi-stage integration of the ATS block into vision transformer models. In the single-stage model, we integrate our ATS module into the stage 3 of DeiT-S. In the multi-stage model, we integrate our ATS module into the stages 3 to 11 of DeiT-S. As it can be seen in Figure 7.8, the multi-stage DeiT-S+ATS performs better than the single-stage DeiT-S+ATS. This is due to the fact that a multi-stage DeiT-S+ATS model can gradually decrease the GFLOPs by discarding fewer tokens in earlier stages, while a single-stage DeiT-S+ATS model has to discard more tokens in earlier stages to reach the same GFLOPs level.

7.3.1.8 ATS Placement

To evaluate the effect of our ATS module's location within a vision transformer model, we add it to different stages of the DeiT-S network and evaluate it on the ImageNet validation set without finetuning the model. To have a better comparison, we set the average computation costs of all experiments to 3 GFLOPs. As it can be seen in Table 7.2, integrating the ATS module into the first stage of the DeiT-S model results in a poor top-1 accuracy of 73.1%. On the other hand, integrating the ATS module into stage 3 results in a 78.5% top-1 accuracy. As mentioned before, earlier transformer blocks are more prone to predict noisier attention weights for the classification token. Therefore,

Method	Top-1 acc	GFLOPs
Top-K	78.9	2.9
Inverse Transform Sampling	**79.7**	2.9

Table 7.1: Comparison of the inverse transform sampling approach with the top-K selection. We finetune and test two different versions of the multi-stage DeiT-S+ATS model: with (1) top-K token selection and (2) inverse transform token sampling. We report the top-1 accuracy of both networks on the ImageNet validation set. For the model with the top-K selection approach, we set $K_n = 0.865 \times \#InputTokens_n$ where n is the stage index. For example, $K_3 = 171$ in stage 3.

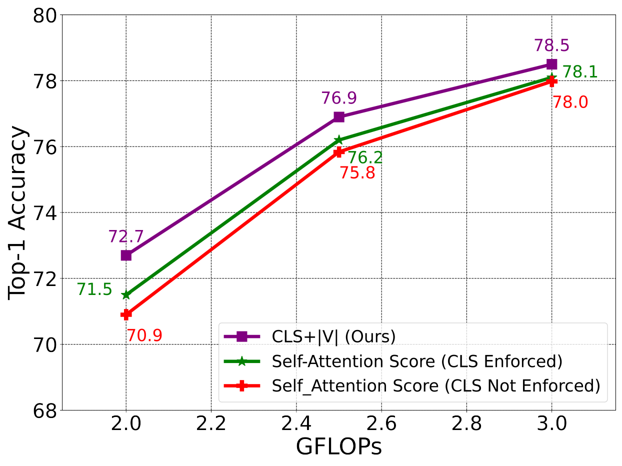


Figure 7.11: Impact of allowing ATS to discard the classification token on the network's accuracy. The model is a single stage DeiT-S+ATS without finetuning.

integrating our ATS module into the first stage performs worse than incorporating it into the stage 3. Although the attention weights of the stage 6 are less noisy, we have to discard more tokens to reach the desired GFLOPs level of 3. For example in stages 0, 3, and 6, we set K to 130, 108, and 56, respectively. The highest accuracy is obtained when we integrate the ATS module into multiple stages of the DeiT-S model. This is because of the progressive token sampling that occurs in a multi-stage DeiT-S+ATS model. In other words, a multi-stage DeiT-S+ATS network can gradually decrease the GFLOPs by discarding fewer tokens in the earlier stages, while a single-stage DeiT-S+ATS model has to discard more tokens in the earlier stages to reach the same GFLOPs level. We also added the ATS module into all stages, yielding average GFLOPs of 2.6 and 76.9% top-1 accuracy.

Stage(s)	0	3	6	3-11
Top-1 Accuracy	73.1	78.5	77.4	**79.2**

Table 7.2: Evaluating the integration of the ATS module into different stages of DeiT-S (Touvron *et al.*, 2021).

7.3.1.9 Adding ATS to Models with Other Token Pruning Approaches

To better evaluate the performance of our adaptive token sampling approach, we also add our module to the state-of-the-art efficient vision transformer EViT-DeiT-S (*Liang et al.*, 2022). EViT (*Liang et al.*, 2022) introduces a token reorganization method that first identifies the top-K important tokens by computing token attentiveness between the tokens and the classification token and then fuses less informative tokens. Interestingly, our ATS module can also be added to the EViT-DeiT-S model and further decrease the GFLOPs, as shown in Table 7.3. These results demonstrate the superiority of our adaptive token sampling approach compared to static token pruning methods. We integrate our ATS module into stages 4, 5, 7, 8, 10, and 11 of the EViT-DeiT-S backbone and fine-tune them for 10 epochs following our fine-tuning setups on the ImageNet dataset discussed earlier.

Model	Top-1 acc	GFLOPs
EViT-DeiT-S (30 Epochs) *Liang et al.* (2022)	79.5	3.0
EViT-DeiT-S (30 Epochs)+ATS	79.5	**2.5**
EViT-DeiT-S (100 Epochs) *Liang et al.* (2022)	79.8	3.0
EViT-DeiT-S (100 Epochs)+ATS	79.8	**2.5**

Table 7.3: Evaluating the EViT-DeiT-S (*Liang et al.*, 2022) model's performance when integrating the ATS module into it with $K_n = 0.7 \times \#InputTokens_n$ where n is the stage index.

7.3.1.10 The Effect of K

In Figures 7.6, 7.7, and 7.8 we varied the value of K to achieve different GFLOPs levels (Top-1 Accuracy vs. GFLOPs). In Figure 7.12, we study the effect of varying K in the ATS module of the single-stage DeiTS+ATS model with fine-tuning. Interestingly, even sampling only 48 tokens (2 GFLOPs) achieves 75% accuracy.

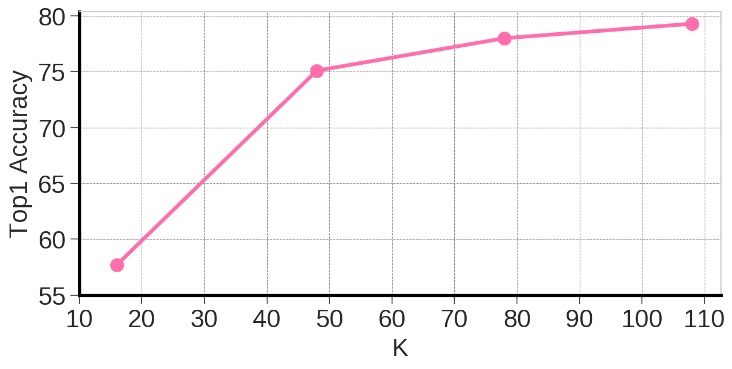


Figure 7.12: We varied the value of K in the ATS module to study the effect of K on the top-1 accuracy. K=48 corresponds to 2 GFLOPs. The backbone model is DeiT-S pre-trained on ImageNet-1K.

Model	Params (M)	GFLOPs	Throughput	Top-1
Deit-S *Touvron et al.* (2021)	22.05	4.6	1010	79.8
Deit-S+ATS	22.05	**2.9**	**1403**	79.7

Table 7.4: We run the models on a single RTX6000 GPU (CUDA 11.0, PyTorch 1.8, image size: 224×224). We average the value of throughput over 20 runs. We add ATS to multiple stages of the DeiT-S model and fine-tune the network on the ImageNet dataset.

Model	Top-1	GFLOPs
XViT+ATS Not-Finetuned(16×)	83.4	**521**
XViT+ATS Finetuned(16×)	84.4	**521**
XViT(16×)	**84.5**	850

Table 7.5: Our ATS module is added to XViT *Bulat et al.* (2021) pre-trained on Kinetics-600.

7.3.1.11 ATS Integration Without Further Training

One of the most important aspects of our approach is that it can be added to any pre-trained off-the-shelf vision transformer. For example, our not fine-tuned multi-stage DeiT-S+ATS model (Figure 7.8) has only a 0.6% (Table 7.6) top-1 accuracy drop while it has improved the efficiency by about 1.6 GFLOPs without any further training of the backbone model. We also observe the same performance on video data. As reported in Table 7.5, our not fine-tuned XViT+ATS model has only a 1.1% top-1 accuracy drop while it has improved the efficiency by about 329 GFLOPs without any further training of the backbone model. This capability of our ATS module roots back in its adaptive inverse transform sampling strategy. Our ATS module samples informative tokens based on their contributions to the classification token. Uninformative tokens that only slightly contribute to the final prediction receive lower attention weights for the classification token. Therefore, the output classification token will be only marginally affected by removing such redundant tokens. On the other hand, the redundant tokens are less similar to the informative tokens and receive lower attention weights for those tokens in the attention matrix. Consequently, they do not contribute much to the value of informative tokens and eliminating them does not change the way informative tokens are contributing to the output classification token.

7.3.1.12 Attention Map Visualization

As shown in Figure 7.13, the attention maps become more focused on the birds and less on the background at the later stages, which is aligned with our observations on the sampled tokens at each stage.

7.3.1.13 Integrating ATS into a Transformer Block

Unlike a standard transformer block in vision transformers, we assign a score to each token and use inverse transform sampling to prune the rows of the attention matrix $\mathcal{A}$ to get $\mathcal{A}^s$. Next, we get the output $\mathcal{O} = \mathcal{A}^s\mathcal{V}$ and forward it to the Feed-Forward Network (FFN) of the transformer block. We

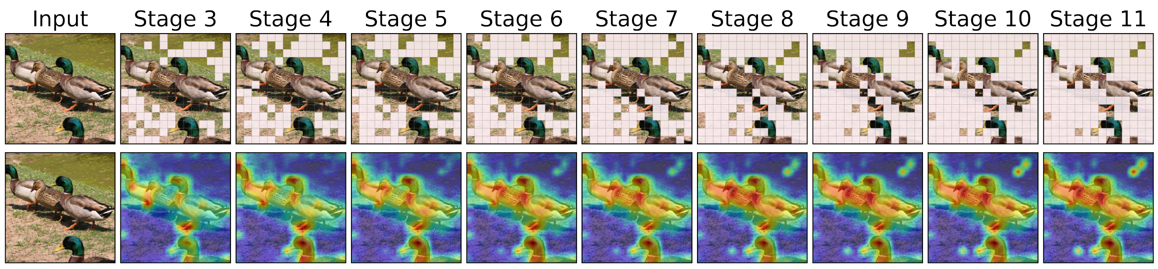


Figure 7.13: Visualization of the sampled tokens and attention maps of a not fine-tuned multi-stage DeiT-S+ATS.

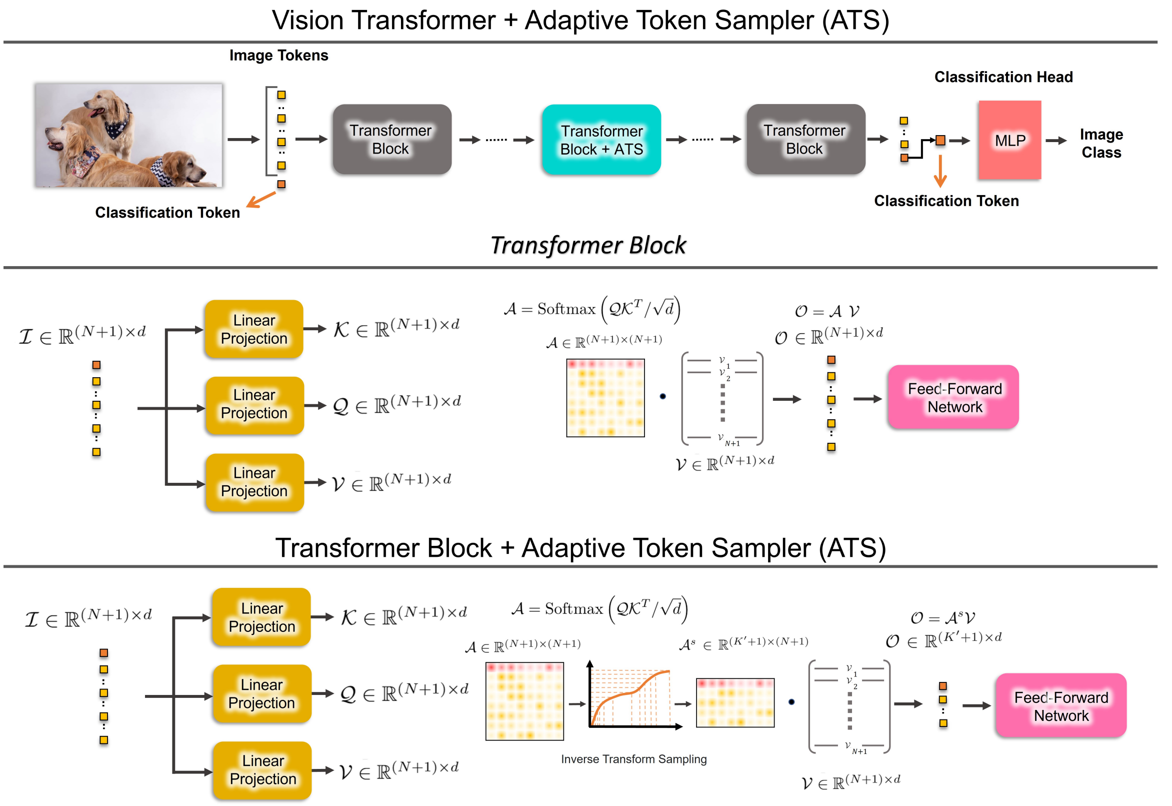

Vision Transformer + Adaptive Token Sampler (ATS)
Image Tokens
Transformer Block
Transformer Block + ATS
Transformer Block
Classification Head
MLP
Image Class
Classification Token
Classification Token
Transformer Block
$\mathcal{I} \in \mathbb{R}^{(N+1)\times d}$
Linear Projection
$\mathcal{K} \in \mathbb{R}^{(N+1)\times d}$
$\mathcal{Q} \in \mathbb{R}^{(N+1)\times d}$
$\mathcal{V} \in \mathbb{R}^{(N+1)\times d}$
$\mathcal{A} = \mathrm{Softmax}\left(\mathcal{Q}\mathcal{K}^T/\sqrt{d}\right)$
$\mathcal{A} \in \mathbb{R}^{(N+1)\times(N+1)}$
$\mathcal{O} = \mathcal{A}\,\mathcal{V}$
$\mathcal{O} \in \mathbb{R}^{(N+1)\times d}$
Feed-Forward Network
Transformer Block + Adaptive Token Sampler (ATS)
$\mathcal{A}^s \in \mathbb{R}^{(K'+1)\times(N+1)}$
Inverse Transform Sampling
$\mathcal{O} = \mathcal{A}^s\mathcal{V}$
$\mathcal{O} \in \mathbb{R}^{(K'+1)\times d}$

Figure 7.14: The Adaptive Token Sampler (ATS) can be integrated into the self-attention layer of any transformer block of a vision transformer model (top). The ATS module takes at each stage a set of input tokens $\mathcal{I}$. The first token is considered as the classification token in each block of the vision transformer. The attention matrix $\mathcal{A}$ is then calculated by the dot product of the queries $\mathcal{Q}$ and keys $\mathcal{K}$, scaled by $\sqrt{d}$. Having selected the significant tokens, we then sample the corresponding attention weights (rows of the attention matrix $\mathcal{A}$) to get $\mathcal{A}^s$. Finally, we softly downsample the input tokens $\mathcal{I}$ to output tokens $\mathcal{O}$ using the dot product of $\mathcal{A}^s$ and $\mathcal{V}$. Next, we forward the output tokens $\mathcal{O}$ through a Feed-Forward Network (FFN) to get the output of the transformer block.

visualize the details of our ATS module, which is integrated into a standard self-attention layer in

Figure 7.14.

7.3.2 Comparison with state-of-the-art

We compare the performances of our adaptive models, which are equipped with the ATS module, with state-of-the-art vision transformers for image and video classification on the ImageNet-1K *Deng et al.* (2009) and Kinetics *Kay et al.* (2017); *Carreira et al.* (2018) datasets, respectively. Tables 7.6-7.8 show the results of this comparison. For the image classification task, we incorporate our ATS module into the stages 3 to 11 of the DeiT-S (*Touvron et al.*, 2021) model. We also integrate our ATS module into the 1^{st} to 9^{th} blocks of the 3^{rd} stage of CvT-13 and CvT-21 (*Wu et al.*, 2021), and into stages 1-9 of the transformer module of PS-ViT (*Yue et al.*, 2021). We fine-tune the models on the ImageNet-1K training set. We also evaluate our ATS module for action recognition. To this end, we add our module to the XViT (*Bulat et al.*, 2021) and TimeSformer (*Bertasius et al.*, 2021) video vision transformers. For more details, please refer to the supplementary materials.
Image Classification As it can be seen in Table 7.6, our ATS module decreases the GFLOPs of all vision transformer models without adding any extra parameters to the backbone models. For the DeiT-S+ATS model, we observe a 37% GFLOPs reduction with only 0.1% reduction of the top-1 accuracy. For the CvT+ATS models, we can also observe a GFLOPs reduction of about 30% with $0.1-0.2\%$ reduction of the top-1 accuracy. More details on the efficiency of our ATS module can be found in the supplementary materials (*e.g.* throughput). Comparing ATS to DynamicViT (*Rao et al.*, 2021a) and HVT (*Pan et al.*, 2021b), which add additional parameters to the model, our approach achieves a better trade-off between accuracy and GFLOPs. Our method also outperforms the EViT-DeiT-S (*Liang et al.*, 2022) model trained for 30 epochs without adding any extra trainable parameters to the model. We note that the EViT-DeiT-S model can improve its top-1 accuracy by around 0.3% when it is trained for much more training epochs (*e.g.* 100 epochs). For a fair comparison, we have considered the 30 epochs training setup used by Dynamic-ViT (*Rao et al.*, 2021a). We have also added our ATS module to the PS-ViT network (*Yue et al.*, 2021). As it can be seen in Table 7.6, although PS-ViT has drastically lower GFLOPs compared to its counterparts, its GFLOPs can be further decreased by incorporating ATS in it.
Action Recognition As it can be seen in Tables 7.7 and 7.8, our ATS module drastically decreases the GFLOPs of all video vision transformers without adding any extra parameters to the backbone models. For the XViT+ATS model, we observe a 39% GFLOPs reduction with only 0.2% reduction of the top-1 accuracy on Kinetics-400 and a 38.7% GFLOPs reduction with only 0.1% drop of the top-1 accuracy on Kinetics-600. We observe that XViT+ATS achieves similar accuracy as Token-Learner (*Ryoo et al.*, 2021) on Kinetics-600 while requiring $17.6\times$ less GFLOPs. For TimeSformer-L+ATS, we can observe 50.8% GFLOPs reduction with only 0.2% drop of the top-1 accuracy on Kinetics-400. These results demonstrate the generality of our approach that can be applied to both image and video representations.

7.3.3 Runtime

Throughput: While ATS is a super-light module, there is still a small cost associated with I/O operations. For a DeiT-S network with a single ATS stage, the sampling overhead is about 1.5% of the overall computation which is negligible compared to the large savings due to the dropped tokens. To further elaborate on this, we have reported the throughput (images/s) of the DeiT-S model with/without our ATS module in Table 7.4. As it can be seen, the speed-up of our module is aligned with its GFLOPs reduction.

Batch Processing: While for most applications the inference is performed for a single image or video, ATS can also be used for inference with a mini-batch. To this end, we rearrange the tokens of each image so that the sampled tokens are in the lower indices. Then, we remove the last tokens completely to reduce the computation. This way, we only process m tokens, where $m = \max_i(K'_i + 1)$ over all images i of the mini-batch. In the worst case scenario (*e.g.* a very large minibatch), we will keep all $K + 1$ first tokens after rearrangement. This will still reduce the computation by a factor of $\frac{N+1}{K+1}$. For example, using a mini-batch of size 512 on the ImageNet validation set, m is 129 in Stage 7 of the DeiT-S+ATS model, which is smaller than the total number of tokens (197). Therefore, we discard at least 68 tokens in stage 7 even in a mini-batch setting. Moreover, for the fully connected layers in a transformer block, which requires most of the computation (*Marin et al.*, 2021), we can flatten the mini-batch dimension and forward only non-zero tokens of the whole mini-batch in parallel through the fully connected layers.

7.3.4 Implementation Details

In our experiments for image classification, we use the ImageNet (**?**) dataset with 1.28M training images and 1K classes. We evaluate our adaptive models, which are equipped with the ATS module, on 50K validation images of this dataset. In our experiments for action recognition, we use the Kinetics-400 (*Kay et al.*, 2017) and Kinetics-600 (*Carreira et al.*, 2018) datasets containing short clips (typically 10 seconds long) sampled from YouTube. Kinetics-400 and Kinetics-600 consist of 400 and 600 classes, respectively. The versions of Kinetics-400 and Kinetics-600 used in this work consist of approximately 261k and 457k clips, respectively. Note that these numbers are lower than the original datasets due to the removal of certain videos from YouTube. Our networks for image classification are trained on 8 NVIDIA Quadro RTX 6000 GPUs and for action recognition on 8 NVIDIA A100 GPUs.

7.3.4.1 DeiT + ATS

Training To fine-tune our adaptive models, we follow the DynamicViT (*Rao et al.*, 2021a) training settings. We use the DeiT model's pre-trained weights to initialize the backbones of our adaptive network and train it for 30 epochs using AdamW optimizer. The learning rate and batch size are set to 5e-4 and 8×96, respectively. We use the cosine scheduler to train the networks. For both multi and single stage models, we set $K = 197$ during training.

Evaluation We use the same setup as (*Touvron et al.*, 2021) for evaluating our adaptive models. To evaluate the performance of our multi-stage DeiT-S+ATS model with different average GFLOPs levels of 3, 2.5, and 2, we set $K_n = \max(\rho \times \#InputTokens_n, 8)$ in which ρ is set to $1, 0.87, 0.72$, respectively, and n is the stage index. For the single-stage model, we set $K = 108, 78, 48$ to evaluate the model with different average GFLOPs levels of 3, 2.5, and 2.

7.3.4.2 CvT + ATS

We integrate our ATS module into the 1^{st} to 9^{th} blocks of the 3^{rd} stage of the CvT-13 and CvT-21 (*Wu et al.*, 2021) networks. For both CvT models, we do not use any convolutional projection layers in the transformer blocks of stage 3.

Training To train our adaptive models, we follow most of the CvT (*Wu et al.*, 2021) network's training settings. We use the CvT model's pre-trained weights to initialize the backbones of our adaptive networks and train them for 30 epochs using AdamW optimizer. The learning rate and batch size are set to 1.5e-6 and 128, respectively. We use the cosine scheduler to train the networks.

Evaluation To evaluate our CvT+ATS model, we use the same setup as *Wu et al.* (2021).

7.3.4.3 PS-ViT + ATS

Training To fine-tune our adaptive models, we follow the PS-ViT (*Yue et al.*, 2021) training settings. We use the PS-ViT model's pre-trained weights to initialize the backbones of our adaptive network and train it for 30 epochs using AdamW optimizer. The learning rate and batch size are set to 5e-4 and 8×96, respectively. We use the cosine scheduler to train the networks.

Evaluation To evaluate our CvT+ATS model, we use the same setup as *Yue et al.* (2021).

7.3.4.4 XViT + ATS

We integrate our ATS module into the stages 3 to 11 of the XViT (*Bulat et al.*, 2021) network.

Training To train our adaptive model, we follow most of the XViT (*Bulat et al.*, 2021) network's training settings. We use the XViT model's pre-trained weights to initialize the backbone of our adaptive network and train it for 10 epochs using SGD optimizer. The learning rate and batch size are set to 1.5e-6 and 64, respectively. We use the cosine scheduler to train the networks.

Evaluation To evaluate our XViT+ATS model, we use the same setup as *Bulat et al.* (2021).

7.3.4.5 TimeSformer + ATS

We integrate our ATS module into the stages 3 to 5 of the TimeSformer (*Bertasius et al.*, 2021) network.

Training To train our adaptive model, we follow most of the TimeSformer (*Bertasius et al.*, 2021) network's training settings. We use the TimeSformer model's pre-trained weights to initialize the backbones of our adaptive networks and train it for 5 epochs using SGD optimizer. The learning rate and batch size are set to 5e-6 and 32, respectively. We use the cosine scheduler to train the networks.

Evaluation To evaluate our TimeSformer-HR+ATS and TimeSformer-L+ATS models, we use the same setup as *Bertasius et al.* (2021).

7.4 Summary

In this dissertation, we initially proposed an innovative spatio-temporal modeling approach to enhance the accuracy of 3D Convolutional Neural Networks (CNNs) for action recognition. Furthermore, we demonstrated that the performance of 3D CNNs can be augmented by utilizing effective pre-training via knowledge transfer from pre-trained models on image classification tasks. We expanded our exploration of video understanding methodologies by broadening the problem scope beyond action recognition. We illustrated that incorporating additional categories besides human actions into the training data not only is feasible, but also enhances the representation learning capacity of the video understanding model. We delved deeper into the study of 3D CNN models by examining their efficiency. To this end, we introduced a novel approach for making 3D CNNs more efficient by adaptively adjusting their computational complexity according to the intricacy of the input video. We substantiate the efficacy of adaptively modifying temporal feature resolutions in 3D CNNs for improved efficiency.

In this chapter, we presented an alternative innovative method aimed at optimizing the efficiency of video understanding models, specifically for vision transformers. To achieve this, we proposed a novel differentiable parameter-free module called the Adaptive Token Sampler (ATS) to enhance the efficiency of vision transformers for image and video classification. The innovative ATS module identifies the most informative and distinctive tokens within the stages of a vision transformer model, ensuring that an optimal number of tokens are utilized for each input image or video clip. By integrating our ATS module into the attention layers of existing vision transformers, which employ a static number of tokens, we can transform them into considerably more efficient vision transformers with an adaptive token count. We demonstrated that our ATS module can be added to off-the-shelf pre-trained vision transformers as a plug-and-play module, subsequently reducing their GFLOPs without necessitating additional training. Moreover, it is feasible to train a vision transformer equipped with the ATS module due to its differentiable design. We evaluated our approach on the ImageNet-1K image recognition dataset and incorporate our ATS module into three distinct state-of-the-art vision transformers. Furthermore, we validated the generality of our approach by incorporating it into various state-of-the-art video vision transformers and assessing their performance on the Kinetics-400 and Kinetics-600 datasets. The results reveal that the ATS module decreases the computational cost (GFLOPs) between 27% and 50.8%, with only a negligible loss in accuracy. Although our experiments predominantly focus on image and video vision transformers, we posit that our approach may also be applicable in other domains, such as audio.

Despite the multitude of methods achieving state-of-the-art results in video understanding on trimmed video clips, real-world scenarios frequently involve untrimmed videos containing multiple actions that necessitate recognition at each frame. This task, known as temporal action segmentation, entails not only identifying actions within untrimmed videos but also ascertaining their sequence and duration. While interest in temporal action segmentation has grown significantly, annotating each frame of a video remains a laborious and expensive process. Consequently, weakly supervised approaches have been developed to learn temporal action segmentation from videos that possess only weak labels. In the subsequent chapters, we investigate weakly supervised temporal action segmentation and propose two novel approaches for temporal action segmentation with varying degrees of weak supervision.

Model	Params (M)	GFLOPs	Resolution	Top-1
ViT-Base/16 *Dosovitskiy et al.* (2021)	86.6	17.6	224	77.9
HVT-S-1 *Pan et al.* (2021b)	22.09	2.4	224	78.0
IA-RED^2 *Pan et al.* (2021a)	-	2.9	224	78.6
DynamicViT-DeiT-S (30 Epochs) *Rao et al.* (2021a)	22.77	2.9	224	79.3
EViT-DeiT-S (30 epochs) *Liang et al.* (2022)	22.1	3.0	224	79.5
DeiT-S+ATS (Ours)	**22.05**	**2.9**	224	79.7
DeiT-S *Touvron et al.* (2021)	22.05	4.6	224	79.8
PVT-Small *Wang et al.* (2021)	24.5	3.8	224	79.8
CoaT Mini *Xu et al.* (2021)	10.0	6.8	224	80.8
CrossViT-S *Chen et al.* (2021)	26.7	5.6	224	81.0
PVT-Medium *Wang et al.* (2021)	44.2	6.7	224	81.2
Swin-T *Liu et al.* (2021)	29.0	4.5	766	81.3
T2T-ViT-14 *Yuan et al.* (2021)	22.0	5.2	224	81.5
CPVT-Small-GAP *Chu et al.* (2021)	23.0	4.6	817	81.5
CvT-13 *Wu et al.* (2021)	20.0	4.5	224	81.6
CvT-13+ATS (Ours)	20.0	**3.2**	224	81.4
PS-ViT-B/14 *Yue et al.* (2021)	21.3	5.4	224	81.7
PS-ViT-B/14+ATS (Ours)	21.3	**3.7**	224	81.5
RegNetY-8G *Radosavovic et al.* (2020)	39.0	8.0	224	81.7
DeiT-Base/16 *Touvron et al.* (2021)	86.6	17.6	224	81.8
CoaT-Lite Small *Xu et al.* (2021)	20.0	4.0	224	81.9
T2T-ViT-19 *Yuan et al.* (2021)	39.2	8.9	224	81.9
CrossViT-B *Chen et al.* (2021)	104.7	21.2	224	82.2
T2T-ViT-24 *Yuan et al.* (2021)	64.1	14.1	224	82.3
PS-ViT-B/18 *Yue et al.* (2021)	21.3	8.8	224	82.3
PS-ViT-B/18+ATS (Ours)	21.3	**5.6**	224	82.2
CvT-21 *Wu et al.* (2021)	32.0	7.1	224	82.5
CvT-21+ATS (Ours)	32.0	**5.1**	224	82.3
TNT-B *Han et al.* (2021)	66.0	14.1	224	82.8
RegNetY-16G *Radosavovic et al.* (2020)	84.0	16.0	224	82.9
Swin-S *Liu et al.* (2021)	50.0	8.7	224	83.0
CvT-13$_{384}$ *Wu et al.* (2021)	20.0	16.3	384	83.0
CvT-13$_{384}$+ATS (Ours)	20.0	**11.7**	384	82.9
Swin-B *Liu et al.* (2021)	88.0	15.4	224	83.3
LV-ViT-S *Jiang et al.* (2021)	26.2	6.6	224	83.3
CvT-21$_{384}$ *Wu et al.* (2021)	32.0	24.9	384	83.3
CvT-21$_{384}$+ATS (Ours)	32.0	**17.4**	384	83.1

Table 7.6: Comparison of the multi-stage ATS models with state-of-the-art image classification models with comparable GFLOPs on the ImageNet validation set. We equip DeiT-S (*Touvron et al.*, 2021), PS-ViT (*Yue et al.*, 2021), and variants of CvT (*Wu et al.*, 2021) with our ATS module and fine-tune them on the ImageNet training set.

Model	Top-1	Top-5	Views	GFLOPs
bLVNet *Fan et al.* (2019)	73.5	91.2	3×3	840
STM *Jiang et al.* (2019)	73.7	91.6	-	-
TEA *Li et al.* (2020b)	76.1	92.5	10×3	2,100
TSM R50 *Lin et al.* (2019)	74.7	-	10×3	650
I3D NL *Wang et al.* (2018c)	77.7	93.3	10×3	10,800
CorrNet-101 *Wang et al.* (2020a)	79.2	-	10×3	6,700
ip-CSN-152 *Tran et al.* (2019)	79.2	93.8	10×3	3,270
HATNet (chapter 5)	79.3	-	10	458
HATNet+ATFR (chapter 6)	79.3	-	10	288
SlowFast 16×8 R101+NL *Feichtenhofer et al.* (2019)	79.8	93.9	10×3	7,020
X3D-XXL *Feichtenhofer* (2020)	80.4	94.6	10×3	5,823
TimeSformer-L *Bertasius et al.* (2021)	80.7	94.7	1×3	7,140
TimeSformer-L+ATS (Ours)	80.5	94.6	1×3	**3,510**
ViViT-L/16×2 *Arnab et al.* (2021)	80.6	94.7	4×3	17,352
MViT-B, 64×3 *Fan et al.* (2021b)	81.2	95.1	3×3	4,095
X-ViT (16×) *Bulat et al.* (2021)	80.2	94.7	1×3	425
X-ViT+ATS (16×) (Ours)	80.0	94.6	1×3	**259**
Swin-L(384) *Liu et al.* (2022)	84.9	96.6	10×5	25,284
TokenLearner 16at18 (L/10) *Ryoo et al.* (2021)	85.4	-	12	49,200

Table 7.7: Comparison with state-of-the-art on Kinetics-400. In accordance with *Bulat et al.* (2021), GFLOPs are calculated as the product of the backbone GFLOPs and the number of views. It is crucial to emphasize that the research presented in this chapter has been published in ECCV 2022 (*Fayyaz et al.*, 2022). To provide a thorough overview and foster a deeper understanding of the prevailing trends in the field, we have integrated contemporary state-of-the-art methods published at or subsequent to the publication date of this chapter.

Model	Top-1	Top-5	Views	GFLOPs
AttentionNAS *Wang et al.* (2020b)	79.8	94.4	-	1,034
LGD-3D R101 *Qiu et al.* (2019)	81.5	95.6	10×3	-
HATNET (chapter 5)	81.6	-	10	458
HATNET+ATFR (chapter 6)	81.6	-	10	288
SlowFast R101+NL *Feichtenhofer et al.* (2019)	81.8	95.1	10×3	3,480
X3D-XL *Feichtenhofer* (2020)	81.9	95.5	10×3	1,452
X3D-XL+ATFR (chapter 6)	82.1	95.6	10×3	768
TimeSformer-HR *Bertasius et al.* (2021)	82.4	96	1×3	5,110
TimeSformer-HR+ATS (Ours)	82.2	96	1×3	**3,103**
ViViT-L/16x2 *Arnab et al.* (2021)	82.5	95.6	4×3	17,352
MViT-B-24, 32×3 *Fan et al.* (2021b)	84.1	96.5	1×5	7,080
TokenLearner 16at12(L/16) *Ryoo et al.* (2021)	84.4	96.0	4×3	9,192
X-ViT (16×) *Bulat et al.* (2021)	84.5	96.3	1×3	850
X-ViT+ATS (16×) (Ours)	84.4	96.2	1×3	**521**
Swin-L(384) *Liu et al.* (2022)	85.9	97.1	4×3	25,284
TokenLearner 16at18 w. Fuser (L/16) *Ryoo et al.* (2021)	86.3	97.0	12	49,200

Table 7.8: Comparison with state-of-the-art on Kinetics-600. In accordance with *Bulat et al.* (2021), GFLOPs are calculated as the product of the backbone GFLOPs and the number of views. It is crucial to emphasize that the research presented in this chapter has been published in ECCV 2022 (*Fayyaz et al.*, 2022). To provide a thorough overview and foster a deeper understanding of the prevailing trends in the field, we have integrated contemporary state-of-the-art methods published at or subsequent to the publication date of this chapter.

CHAPTER 8

Weakly Supervised Temporal Action Segmentation from Action Sets

This chapter is based on the following publication:
SCT: Set Constrained Temporal Transformer for Set Supervised Action Segmentation
Mohsen Fayyaz and Juergen Gall
IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2020.

Below we will summarize the contributions of each author.

- **Mohsen Fayyaz** was responsible for proposing, implementing, and evaluating the method presented in this work. He also made significant contributions to the writing of the paper
- **Juergen Gall** played an instrumental role in the success of this project through his invaluable supervision and guidance. Furthermore, his contributions through suggestions, discussions, writing, and funding the project were essential to the completion of this work.

Contents

In the preceding chapters, we presented methods that achieved cutting-edge results in video understanding for trimmed video clips. Nonetheless, real-world situations frequently involve untrimmed videos containing multiple actions, necessitating recognition at each frame. This challenge, known as temporal action segmentation, entails not only identifying actions within untrimmed videos but also ascertaining their sequence and duration. Despite the growing interest in this area, annotating each frame of a video remains a laborious and expensive endeavor. Consequently, weakly supervised techniques have emerged to learn temporal action segmentation from videos with only minimal labeling. In such instances, identifying the set of actions present in a video serves as weak supervision, offering a cost-effective alternative for video annotation. In this chapter, we propose an end-to-end trainable approach to train a temporal action segmentation model with such weak supervision. Given a set of actions in which only the list of actions occurring in the video is known, but not when, how often, or in which order they occur, our approach divides the video into smaller temporal regions and predicts for each region the action label and its length, as well as estimating action labels for each frame. By measuring the consistency of the frame-wise predictions with respect to the temporal regions and annotated action labels, the network learns to divide the video into class-consistent regions. To demonstrate the efficacy of our method, we evaluate our approach on two datasets and attain superior performance in comparison to previous approaches.

8.1 Introduction

For many applications, a large amount of video data needs to be analyzed. This includes temporal action segmentation, which requires labeling each frame in a long video by an action class. In the last years, several strong models for temporal action segmentation have been proposed (*Kuehne et al.*, 2016b; *Lea et al.*, 2017; *Abu Farha and Gall*, 2019). These models are, however, trained in a fully supervised setting, *i.e.*, each training video needs to be fully annotated by frame-wise labels. Since acquiring such annotations is very expensive, several works investigated methods to learn the models with less supervision. An example of weakly annotated training data are videos where only transcripts are provided (*Kuehne et al.*, 2017; *Huang et al.*, 2016; *Richard et al.*, 2017, 2018b; *Ding and Xu*, 2018; *Chang et al.*, 2019; *Souri et al.*, 2022; *Li et al.*, 2019). While transcripts of videos can be obtained from scripts or subtitles, they are still costly to obtain. *Richard et al.* (2018a) therefore proposed to learn temporal action segmentation only from a set of action labels that are provided for a complete video of several minutes. In this case, it is only known which actions occur, but not when, in which order, or how often. This makes the task much more challenging compared to learning from transcripts or fully supervised learning.

In the study conducted by *Richard et al.* (2018a), the problem has been addressed by hypothesizing transcripts that contain each action label of a video at least once and then infer a frame-wise labeling of the video by aligning the hypothesized transcripts. While the approach showed that it is possible to learn from such weak annotation even for long videos, the approach does not solve the problem directly but converts it into a weakly supervised learning problem where multiple hypothesized transcripts per video are given. This is, however, ineffective since it is infeasible to align all transcripts that can be generated from a set of action labels and it uses the provided annotations not directly for learning.

In this work, we propose a method that uses the action labels that are given for each training video directly for the loss function. In this way, we can train the model in an end-to-end fashion. The main idea is to divide a video into smaller temporal regions as illustrated in Figure 8.1. For each region, we estimate its length and the corresponding action label. Since for each training video the set of actions is known, we can directly apply a set loss to the predicted action labels of the temporal regions, which penalizes the network if it predicts actions that are not present in the video or if it misses an action. The problem, however, is that we cannot directly apply a loss to the prediction of the region lengths. While a regularizer for the predicted length that penalizes if the lengths of the regions get too large improves the results, it is insufficient as we show in our experiments. We therefore introduce a second branch to make frame-wise predictions and measure how consistent the frame-wise predictions are with respect to the temporal regions and the annotated action labels. Using our differentiable Set Constrained Temporal Transformation (SCT), this loss affects the lengths of the regions, which substantially improves the accuracy of the model.

In our experimental evaluation on three datasets, we show that the proposed approach achieves superior results. We furthermore thoroughly evaluate the impact of each component.

8.2 Weakly Supervised Temporal Action Segmentation

Action segmentation requires to temporally segment all frames of a given video, *i.e.*, predicting the action in each frame of a video. The task can be formulated as follows. Given an input sequence of D-dimensional features $X_{1:T} = (x_1, \dots, x_T)$, $x_t \in \mathbb{R}^D$, the task is to infer the sequence of framewise action labels $\hat{Y}_{1:T} = (\hat{y}_1, \dots, \hat{y}_T)$ where there are C classes $\mathcal{C} = \{1, \dots, C\}$ and $\hat{y}_t \in \mathcal{C}$.

In the case of fully supervised learning, the labels $\hat{Y}_{1:T}$ are provided for each training sequence. In this study, we delve into the weakly supervised paradigm as explored in the seminal work by *Richard et al.* (2018a). In this setting, only the actions $\hat{A} = \{\hat{a}_1, \dots, \hat{a}_M\}$ that occur in a long video are given where $\hat{a}_m \in \mathcal{C}$ and $M \leqslant C$. In contrast to other weakly supervised settings where transcripts are given, this is a much more difficult task since not only the lengths of the actions are unknown for the training sequences, but also the order of the actions and the number of the occurrences of each action.

8.3 Set Supervised Temporal Action Segmentation

In order to address weakly supervised action segmentation, we propose a network that divides a temporal sequence into temporal regions and that estimates for each region the action and the length as illustrated in Figure 8.1. This representation is between a frame-wise representation where the length of each region is just one frame and an action segment representation where a region contains all neighboring frames that have the same action label.
Figure 9.2 illustrates our proposed network, which consists of three components. The first component $f_e(X)$, which is described in Section 8.3.1, maps the input video features $X \in \mathbb{R}^{T \times D}$ to a temporal embedding $Z \in \mathbb{R}^{T' \times D'}$ where $T' < T$ and $D' < D$. The second component $f_r(Z)$, which is described in Section 8.3.2, takes Z as input and estimates for K temporal regions the actions probabilities $A_{1:K} = (a_1, \dots, a_K), a_k \in \mathbb{R}^C$, and the temporal lengths of the regions $L_{1:K} = (\ell_1, \dots, \ell_K), \ell_k \in \mathbb{R}$. In order to obtain the frame-wise class probabilities $Y \in \mathbb{R}^{T \times C}$ from L and A, the third component $f_u(A, L)$, which is discussed in Section 8.3.3, upsamples the estimated

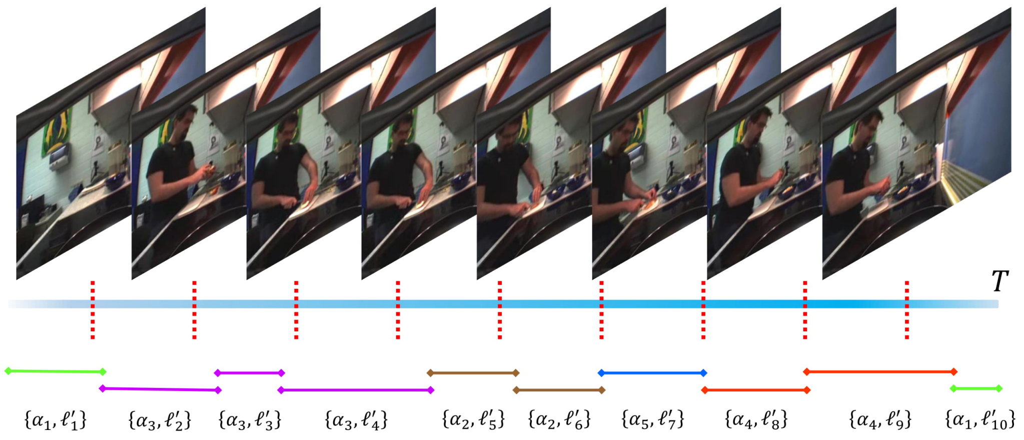


Figure 8.1: Our model estimates for K temporal regions the actions probabilities $A_{1:K} = (a_1, \ldots, a_K), a_k \in \mathbb{R}^C$, and the temporal lengths of the regions $L_{1:K} = (\ell_1, \ldots, \ell_K), \ell_k \in \mathbb{R}$. In this example, $K = 10$. Since temporal regions are not aligned with the action segments, the model estimates the temporal lengths to refine the corresponding temporal region of the predicted action.

regions such that there are T regions of length 1.

8.3.1 Temporal Embedding

Given the input video features $X \in \mathbb{R}^{T \times D}$ the temporal embedding component $f_e(X)$ outputs the hidden video representation $Z \in \mathbb{R}^{T' \times D'}$. Our temporal embedding is a fully convolutional network. In this network we first apply a 1-d convolution with kernel size 1 to reduce the input feature dimension from D to D'. On top of this layer we have used B temporal convolution blocks (TCB) with $B = 6$. Each TCB contains a dilated 1-d convolution layer with kernel size 3 for conducting the temporal structure. We increase the dilation rates as $\{2^b | b \in \mathbb{Z}^+, 0 \leqslant b \leqslant B\}$ where b is the index of the TCBs. Then a ReLU activation function is applied on top of the convolutional layer. On top of this combination, a 1-d convolution layer with kernel size 1 and a residual connection is used. Finally, a dropout with a probability of 0.25 is applied on top. The TCB is modeled after the WaveNet architecture (*van den Oord et al.*, 2016). To reduce the temporal dimension of the representation, we perform temporal max poolings with a kernel size of 2 on top of the TCBs with $b = \{1, 2, 4\}$. Using the TCBs and max poolings, we get large receptive fields on the input data X. Having such large receptive fields provides the capability of modeling long and short range temporal relations between the input frames.

8.3.2 Temporal Regions

On top of the temporal embedding, we use the temporal region estimator network $f_r(Z)$ to estimate the action probabilities and the temporal lengths for K temporal regions. $f_r(Z)$ outputs the hidden representation $Z'_{1:K} = (z'_1, \ldots, z'_K), z'_k \in \mathbb{R}^{D'}$, for the temporal regions. To have a better representation for estimating the actions probabilities A and region lengths L, we increase the receptive field size and decrease the temporal dimension. This network mostly follows the same architecture design as $f_e(X)$. It has B' TCBs with $B' = 4$. The dilation rates of the TCBs are set

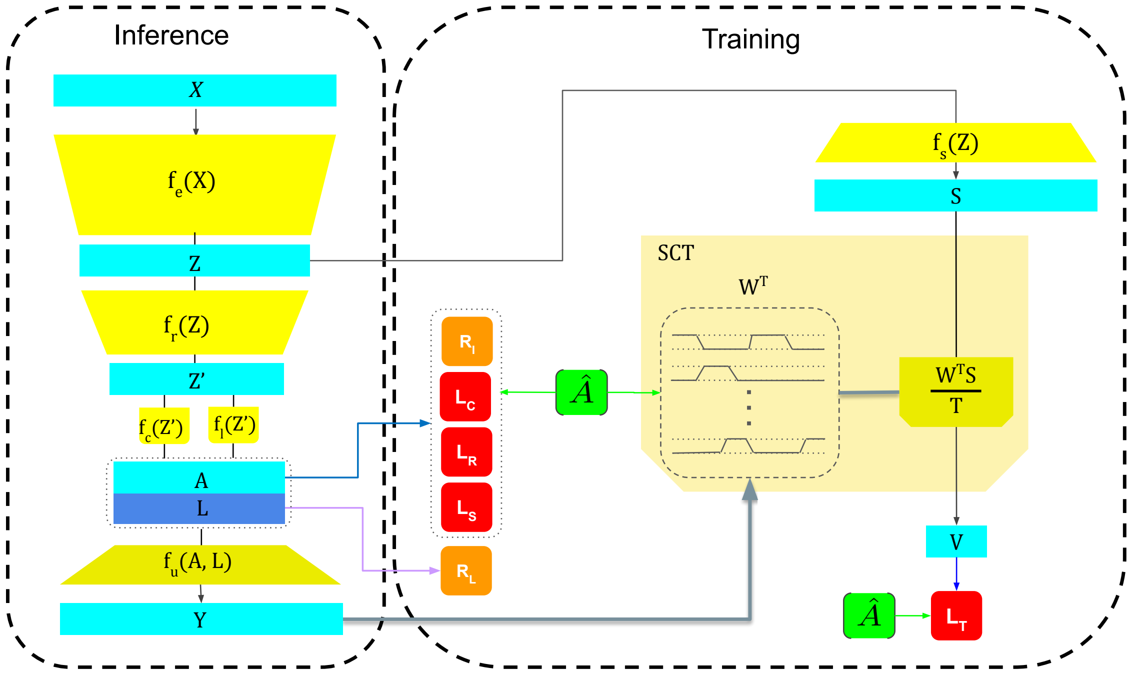


Figure 8.2: Overview of the proposed network with loss functions. The network gets a sequence of features $X_{1:T}$ as input. A temporal model $f_e(X)$ maps these features to a latent space Z with lower temporal resolution. The lower branch $f_r(Z)$ divides the temporal sequence into temporal regions $Z'_{1:K}$ and estimates for each region the action probabilities a_k and the length l_k. Since the temporal resolution has been decreased, the upsampling module $f_u(A, L)$ uses the lengths $L_{1:K}$ and the action probabilities $A_{1:K}$ of all regions to obtain estimates of the framewise probabilities $Y_{1:T}$. While $L_{1:K}$ is regularized by the length regularizer $\mathcal{R}_{\mathcal{L}}$, $A_{1:K}$ is trained to minimize $\mathcal{L}_{\mathcal{S}}$, $\mathcal{L}_{\mathcal{R}}$, $\mathcal{L}_{\mathcal{C}}$, and $\mathcal{R}_{\mathcal{I}}$. Since besides of the regularizer $\mathcal{R}_{\mathcal{L}}$, there is no loss term that provides supervision for L, we use a second branch $f_s(Z)$ to provide an additional supervisory signal. Using SCT, we transform the temporal representations $Y_{1:T}$ and $S_{1:T}$ to a set representation $V_{1:M}$ for the self supervision loss $\mathcal{L}_{\mathcal{T}}$.

as $\{2^{b'} | b' \in \mathbb{Z}^+, B < b' \leqslant B + B'\}$. To reduce the temporal dimension of the representation, we perform temporal max poolings with kernel size 2 on top of the TCBs with indices 2 and 4. On top of the final TCB, we have two different heads f_c and f_l. $f_c(Z')$ predicts the class probabilities A. It consists of a 1-d convolution layer with a kernel size of 1 and an output channel size of C. A softmax function is applied on top of the convolution layer to get the action probabilities A. $f_l(Z')$ predicts the lengths L for the corresponding temporal regions. It consists of two 1-d convolution layers with kernel sizes 1 and output channels $D'/2$ and 1, respectively.

8.3.3 Region Upsampling

$f_c(Z')$ estimates the action probabilities $A_{1:K}$ for temporal regions. To get probabilities for temporal action segmentation, we need to upsample $A_{1:K}$ to $Y_{1:T}$. Since $f_l(Z')$ predicts the corresponding lengths $L_{1:K}$, we can use theses lengths to upsample the probabilities A. To do so, we first project the predicted lengths $L_{1:K}$ to absolute lengths $L'_{1:K} = (\ell'_1, ..., \ell'_K), \ell'_k \in \mathbb{Z}^+$, by:

$$\ell'_k = T \frac{e^{\ell_k}}{\sum_{i=1}^{K} e^{\ell_i}}. \tag{8.1}$$

In other words, we apply the softmax function on L to get the relative lengths, which sum up to 1 and then multiply them by T to get the absolute lengths. Therefore, the absolute lengths sum up to T, which is our desired final temporal size for Y. Given the absolute lengths L', we upsample A in a differentiable way such that $a_k \in \mathbb{R}^C$ becomes $a'_k \in \mathbb{R}^{\ell'_k \times C}$.

8.3.3.1 Temporal Sampling

Although it is possible to obtain a'_k by just copying ℓ'_k times the probabilities a_k, this operation is not differentiable with respect to ℓ'_k. However, we need a differentiable operation in order to update the parameters of f_l, which predicts L, during training.

We first generate our target matrix $a'_k \in \mathbb{R}^{H \times C}$ where $H = \max_k \ell'_k$, *i.e.*, the matrix is set such that the size is constant for all k. For a better temporal sampling, we also expand the source by copying j times a_k, where J is a canonical value equal to 100. Although a_k has been expanded to $\mathbb{R}^{J \times C}$, we still keep the notation a_k.

The idea is to fill the matrix a'_k by backward warping and a bilinear kernel. Similar to *Jaderberg et al.* (2015), we use normalized element indices, such that $-1 \leqslant i_a[j] \leqslant 1$ when $j \in [1 \dots J]$ and $-1 \leqslant i_{a'}[h] \leqslant 1$ when $h \in [1 \dots H]$. This means if we just interpolate the values for each column c, the operation is defined by

$$a'_k[h, c] = \sum_{j=1}^{J} a_k[j, c] \max\left(0, 1 - |i_{a'}[h] - i_a[j]|\right) \tag{8.2}$$

for $h \in [1 \dots H]$.

However, we do not want to fill the entire row but only until ℓ'_k. We therefore apply a 1D affine transformation to the index function

$$T_{\ell'_k}(i_{a'}[h]) = \frac{H}{\ell'_k} i_{a'}[h] + \frac{H}{\ell'_k} - 1. \tag{8.3}$$

This means that $T_{\ell'_k}(i_{a'}[1]) = -1$ and $T_{\ell'_k}(i_{a'}[\ell'_k]) = 1$. By integrating (8.3) into (8.2), we obtain the upsampling operation

$$a'_k[h, c] = \sum_{j=1}^{J} a_k[j, c] \max\left(0, 1 - \left|T_{\ell'_k}(i_{a'}[h]) - i_a[j]\right|\right) \tag{8.4}$$

that is differentiable with respect to ℓ'_k.

Finally, the matrix is cropped to $a'_k \in \mathbb{R}^{\ell'_k \times C}$ and we obtain $Y \in \mathbb{R}^{T \times C}$ by concatenating the a'_ks for $k = 1, \dots, K$.

8.4 Training

In Section 8.3 we proposed a model that is capable of dividing a temporal sequence into temporal regions and predicting corresponding action probabilities A and lengths L. We now discuss the loss functions and regularizers for training the model.

8.4.1 Set Loss

In a set supervised problem we already have the set supervision. So we use a simple set prediction loss $\mathcal{L}_\mathcal{S}$ to use the given set of actions $\hat{A}$. We apply a global max pooling over the temporal dimension of A to output $a^{mc} \in \mathbb{R}^C$. Then we use the binary cross entropy loss for multiclass classification as

$$\mathcal{L}_\mathcal{S} = -\frac{1}{C}\left(\sum_{m \in \hat{A}} \log\left(a^{mc}[m]\right) + \sum_{m \notin \hat{A}} \log\left(1 - a^{mc}[m]\right)\right). \tag{8.5}$$

This loss encourages the model to assign at least one region to one of the classes in $\hat{A}$ and none to the other classes.

8.4.2 Region Loss

The set loss only enforces that there is one region with a high probability for each class. It can, however, happen that the other regions have a uniform distribution for the classes in $\hat{A}$. Since it is unlikely that all actions occur at the same time, we introduce the region loss, which encourages the model to predict only one action from $\hat{A}$ per region. Since we know that only actions from $\hat{A}$ can occur, we first discard the unrelated actions from $A \in \mathbb{R}^{K \times C}$ and denote it by $A^\mathcal{S} \in \mathbb{R}^{K \times M}$, where each column belongs to one of the given action set members $\hat{a}_m$. We now prefer a prediction where for each k the probability is close to 1 for one action $\hat{a}_m$. Due to the softmax, this means that the probability is close to zero for the other actions.

This is achieved by applying a global max pooling over the m dimension of $A^\mathcal{S} \in \mathbb{R}^{K \times M}$ to obtain $a^{mk} \in \mathbb{R}^K$ and using the cross entropy loss:

$$\mathcal{L}_\mathcal{R} = -\frac{1}{K}\sum_{k=1}^{K} \log\left(a^{mk}[k]\right). \tag{8.6}$$

8.4.3 Inverse Sparsity Regularization

The set loss and the region loss ensure that (i) all actions that are not in the set $\hat{A}$ have a low probability, (ii) for each temporal region there is exactly one action $\hat{a}_m \in \hat{A}$ with high probability, and (iii) for each action $\hat{a}_m$ there is at least one region k where $a[k, m]$ is high. This, however, can result in unlikely solutions where for $M - 1$ actions there is only one region with high probability whereas the other regions are assigned to a single action class. To prevent such a sparse distribution of regions for some action classes, we introduce an inverse sparsity regularization term $\mathcal{R}_\mathcal{I}$, which prefers a more balanced class distribution averaged over all regions:

$$\mathcal{R}_\mathcal{I} = \frac{1}{M}\sum_{m \in \hat{A}}\left(1 - \frac{1}{K}\sum_{k=1}^{K} a[k, m]\right). \tag{8.7}$$

This regularizer encourages that the action classes compete for maximizing the number of temporal regions they are being predicted for.

8.4.4 Temporal Consistency Loss

As illustrated in Figure 8.1, the temporal regions are usually smaller than the action segments in the video and a single action often spans several regions. The likelihood of observing the same action in the neighboring temporal regions is therefore usually higher than observing a different action. We therefore introduce the temporal consistency loss $\mathcal{L}_\mathcal{C}$, that encourages the model to predict similar action labels for neighboring temporal regions:

$$\mathcal{L}_\mathcal{C} = \frac{1}{M} \sum_{m \in \hat{A}} \frac{1}{K} \sum_{k=2}^{K} |a[k,m] - a[k-1,m]|. \tag{8.8}$$

More precisely, $\mathcal{L}_\mathcal{C}$ encourages the model to have less prediction changes over the temporal dimension of A^S.

8.4.5 Self Supervision Loss

The aforementioned losses and regularizers only affect the class probabilities A of the temporal regions, but do not backpropagate gradients through the subnetwork f_l. This means that the network does not learn the corresponding lengths of the regions during training. In order to provide an auxiliary supervision signal to train L, we employ a self supervision technique which relies on using two different representations. The first representation Y is obtained by estimating the actions probabilities A and lengths L for K temporal regions as described in Section 8.3.2. Due to the temporal sampling, the representation Y is differentiable with respect to L.

To have another representation, we use a second branch $f_s(Z)$. This branch consists of a subnetwork that has a single 1-d convolution with kernel size 1 and output channel size C. It predicts frame-wise class probabilities for the temporal size T'. We linearly interpolate it to $S \in \mathbb{R}^{T \times C}$ along the temporal dimension, which corresponds to a setting where $K = T'$ and $\ell_k = \frac{T}{T'}$, *i.e.*, all regions have a constant length.

Since we do not know the ground-truth lengths L but only the set of present actions $\hat{A}$, we combine Y and S to compute class probabilities $V_{1:M} = (v_1, ...v_M), v_m \in \mathbb{R}^C$, for each element in the set $\hat{A}$. This is done by the Set Constrained Temporal Transformer module (SCT).

8.4.6 Set Constrained Temporal Transformer

As it is illustrated in Figure 9.2, we produce for each action class $\hat{a}_m \in \hat{A}$ masks w_m from Y. The masks indicate the temporal locations where the action $\hat{a}_m$ occurs in the video. We use these masks to sample from S:

$$v_m = \frac{1}{T} \sum_{t=1}^{T} w_m[t] S[t]. \tag{8.9}$$

If S and Y are consistent, $v_m[\hat{a}_m]$ should be high and $v_m[\hat{a}_n]$ should be close to zero for $n \neq m$.

To exploit this, we apply a softmax on v_m to get the predicted probabilities for the given action and use the cross entropy loss:

$$\mathcal{L}_{\mathcal{T}_m}(v_m, \hat{a}_m) = -\log \left(\frac{e^{v_m[\hat{a}_m]}}{\sum_{c=1}^{C} e^{v_m[c]}} \right). \tag{8.10}$$

Since w_m is differentiable with respect to a_m and l_m, the loss affects both.

As a more efficient way, we can apply all of the masks W on S using:

$$V = \frac{W^T S}{T} \tag{8.11}$$

where $V \in \mathbb{R}^{M \times C}$ and $W^T \in \mathbb{R}^{M \times T}$ denotes the transposed version of W. Therefore, we can define the loss for V and the given actions set $\hat{A}$ as

$$\mathcal{L}_{\mathcal{T}}(V, \hat{A}) = -\frac{1}{M} \sum_{m=1}^{M} \log \left(\frac{e^{V[m,\hat{a}_m]}}{\sum_{c=1}^{C} e^{V[m,c]}} \right). \tag{8.12}$$

8.4.6.1 Backpropagation

Using the $\mathcal{L}_{\mathcal{T}}$ loss, the gradient can backpropagate through both S and Y. Y is the output of $f_u(A, L)$ which is a differential function over L and A. Therefore, we can update the f_l weights using the backpropagated gradients. To be able to backpropagate through the a'_ks we define the gradients with respect to the sampling indices $T_{\ell'_k}(i_{a'}[h])$ as

$$\frac{\partial a'_k[h,c]}{\partial T_{\ell'_k}(i_{a'}[h])} = \\ \sum_{j=1}^{J} a_k[j,c] \begin{cases} 0 & |i_a[j] - T_{\ell'_k}(i_{a'}[h])| \geqslant 1 \\ 1 & i_a[j] - 1 < T_{\ell'_k}(i_{a'}[h]) \leqslant i_a[j] \\ -1 & i_a[j] < T_{\ell'_k}(i_{a'}[h]) < i_a[j] + 1 \end{cases}. \tag{8.13}$$

Since the sampling indices $T_{\ell'_k}(i_{a'}[h])$ are a function of the predicted lengths $L_{1:M}$, the loss gradients are backpropagated to the predicted lengths

8.4.6.2 Region Length Regularization

Learning the lengths based on $\mathcal{L}_{\mathcal{T}}$ may result in degenerated lengths which are close to zero. Therefore, we use a length regularizer $\mathcal{R}_{\mathcal{L}}$ to prevent such circumstances. We define $\mathcal{R}_{\mathcal{L}}$ as

$$\mathcal{R}_{\mathcal{L}} = \frac{1}{K} \sum_{t=1}^{K} (ReLU(\ell_t - \delta) + ReLU(-\ell_t - \delta)) \tag{8.14}$$

where δ is a canonical value equal to 1. This regularization term penalizes the lengths which are bigger or smaller than the length width of δ.

8.4.7 Overall Loss

All of the loss functions and regularizers that we mentioned in this section encourage the model to exploit the given weak supervision and also characteristics of actions to train the model for action segmentation. Therefore, the final loss function for the model is the weighted sum of the above mentioned losses and regularizers. In Section 9.4 we study the impact of all loss functions and regularizers.

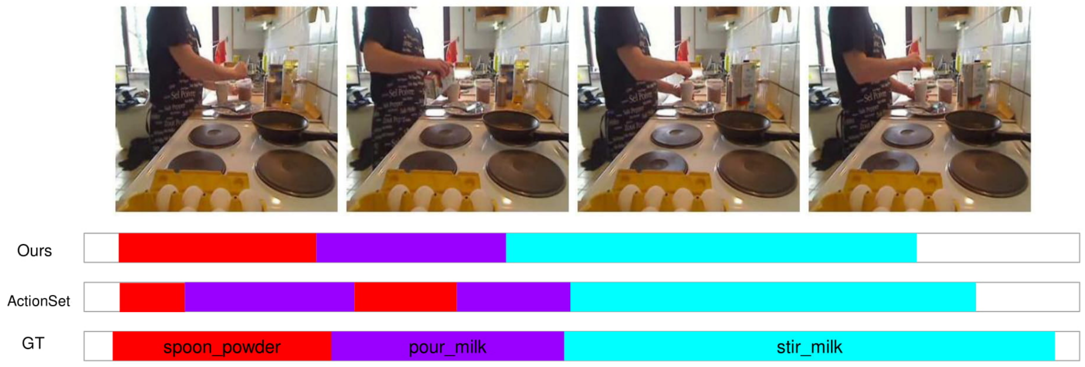


Figure 8.3: Comparing the segmentation quality of our method to the ActionSet method. Our method has predicted the right order of actions occurring in this video. Our approach also estimates the actions lengths better.

8.5 Experiments

In this section, we analyze the components of our approach. We first analyze the model design. Then we evaluate the effect of using different loss functions and regularizers. Finally, we compare our method with the preceding approaches.

8.5.1 Setup

Datasets. We evaluate our method on two popular datasets, namely the Breakfast dataset (*Kuehne et al.*, 2014), and Hollywood Extended *Bojanowski et al.* (2014).

The **Breakfast** dataset contains $1,712$ videos of different cooking activities, corresponding to about 67 hours of videos and 3.6 million frames. The videos belong to 10 different types of breakfast activities like *fried egg* or *coffee* which consist of 48 different fine-grained actions. The actions are densely annotated and only 7% of the frames are background. We report the average frame accuracy (MoF) metric over the predefined train/test splits following *Richard et al.* (2018a).

Hollywood Extended contains 937 video sequences with roughly $800,000$ frames. About 61% of the frames are background, which is comparably large compared to other datasets. The videos contain 16 different action classes. We report the Jaccard index (intersection over union) metric over the predefined train/test splits following *Richard et al.* (2018a).

Feature Extraction. In this study, we employ RGB+flow I3D representations as detailed in *Carreira and Zisserman* (2017), in addition to the XViT($16\times$) and XViT+ATS($16\times$) networks pretrained on the Kinetics dataset *Kay et al.* (2017), to extract features from each frame of the video. To enable a fair comparison, we also utilize the same IDT features as conducted by *Richard et al.* (2018a) and assess the influence of utilizing various features.

Implementation Details. We train all modules of our network together. The hidden size of the temporal embedding module is 128. We use SGD optimizer with weight decay 0.005. The initial learning rate is set to 0.01. Additional details and code are available online.[1]

[1]http://mohsenfayyaz89.github.io

#max poolings	0	1	2	3	4	5	6
MoF	12.3	15.4	20.8	28.1	27.2	21.3	18.2

Table 8.1: Evaluating the effect of changing the numbers of max pooling operations in the temporal embedding module. Experiments are run on Breakfast split 1.

8.5.2 Ablation Experiments

In this section we first analyze the model design. Then we analyze the effect of our loss functions and regularizers on training the model.

8.5.2.1 Effect of different downsampling levels

As mentioned in Section 8.3.1, we downsample the input by applying temporal max pooling with kernel size 2. We evaluate the effect of downsampling by changing the numbers of temporal max pooling operations in the temporal embedding module. It should be mentioned that we apply each max pooling on top of each temporal convolution block (TCB). As can be seen in Table 8.1, a small number of max pooling operations results in a relatively low frame-wise accuracy. This is due to high number of temporal regions, which may result in an over-segmentation problem. Furthermore, a drop in performance can be observed when the number of max pooling operations is too large. In this case, there are not enough temporal regions and a temporal region covers multiple actions. For the rest of the experiments, we use 3 max pooling operations in our temporal modeling module. It should be noted that we use 3 max pooling operations after the TCBs with indices $\{1, 2, 4\}$, while in this experiment 3 max pooling operations are applied after the TCBs with indices $\{1, 2, 3\}$.

8.5.2.2 Effect of using different loss functions and regularizers

As mentioned in Section 8.4, we use different loss functions and regularizers to train our model. To quantify the effect of using these loss functions and regularizers, we train our model with different settings in which we can evaluate the effect of them for training. We train our model on split 1 of the Breakfast dataset and report the results in Table 8.2.

As mentioned in Section 8.4.1, the **Set Loss** $\mathcal{L}_{\mathcal{S}}$ encourages the model to have at least one temporal region with a high class probability for each action in the target action set. Therefore, this loss does not affect the frame-wise prediction performance of the model. As it can be seen in Table 8.2, using only $\mathcal{L}_{\mathcal{S}}$ the model achieves MoF of 8.1%.

By adding the **Region Loss** $\mathcal{L}_{\mathcal{R}}$, we encourage the model to only predict one action per temporal region. Using this auxiliary supervision, the MoF slightly improves to 9.9% which is still relatively low.

As mentioned in Section 8.4.3, adding the **Inverse Sparsity Regularizer** $\mathcal{R}_{\mathcal{I}}$ helps the method to prevent a sparse distribution of regions for some action classes. Therefore, by adding $\mathcal{R}_{\mathcal{I}}$ to the overall loss, the MoF improves to 19.2%, which is significantly better than predicting every frame as background which covers about 7% of the frames.

We further add the **Temporal Consistency Loss** $\mathcal{L}_{\mathcal{C}}$ to encourage the model to predict similar actions for neighboring temporal regions. $\mathcal{L}_{\mathcal{C}}$ improves the result to 21.9%.

As mentioned in Section 8.4.5, all of the aforementioned losses and regularizers only affect the class probabilities A of the temporal regions and do not backpropagate gradients through the length

$\mathcal{L}_\mathcal{S}$	$\mathcal{L}_\mathcal{R}$	$\mathcal{R}_\mathcal{I}$	$\mathcal{L}_\mathcal{C}$	$\mathcal{L}_\mathcal{T}$	$\mathcal{R}_\mathcal{L}$	$\mathcal{L}_\mathcal{J}$	MoF
✓	-	-	-	-	-	-	8.1
✓	✓	-	-	-	-	-	9.9
✓	✓	✓	-	-	-	-	19.2
✓	✓	✓	✓	-	-	-	21.9
✓	✓	✓	✓	✓	-	-	29.9
✓	✓	✓	✓	✓	✓	-	**30.8**
✓	✓	✓	✓	-	✓	-	22.2
✓	✓	✓	✓	-	✓	✓	25.3

Table 8.2: Evaluating the effect of using different losses and regularizers. Experiments are run on Breakfast split 1.

estimator head f_l. We therefore add the **Self Supervision Loss** $\mathcal{L}_\mathcal{T}$ to evaluate the effect of learning lengths during training. Adding $\mathcal{L}_\mathcal{T}$ significantly improves the accuracy to 29.9%. This improvement shows the effect of refining the temporal regions using the predicted lengths.

We also evaluate the effect of using the **Region Length Regularization** $\mathcal{R}_\mathcal{L}$. As mentioned in Section 8.4.6.2, learning the lengths only based on $\mathcal{L}_\mathcal{T}$ may result in too diverse estimated lengths for temporal regions. Therefore, we evaluate the effect of $\mathcal{R}_\mathcal{L}$ by adding it to the overall loss. By adding this regularizer the accuracy improves to 30.8%. Since $\mathcal{L}_\mathcal{T}$ and $\mathcal{R}_\mathcal{L}$ are the only loss function and regularizer which affect the lengths L, we also evaluate the effect of only using $\mathcal{R}_\mathcal{L}$ as an effective regularizer on the lengths without $\mathcal{L}_\mathcal{T}$. This setting results in an MoF of 22.2%. The reason for such a significant drop in performance is that $\mathcal{R}_\mathcal{L}$ only encourages the model to not estimate too diverse lengths. This shows that the proposed self supervision loss based on the set constrained temporal transformer is important to learn proper lengths for the temporal regions.

To have a better understanding of the **Self Supervision Loss**, we also try to train the temporal regions' length estimator head f_l in a different way. Instead of using $\mathcal{L}_\mathcal{T}$, we use the Jensen Shannon Divergence loss which is a symmetric and smoothed version of the KullbackâLeibler divergence to match both representations Y and S as follows:

$$\mathcal{L}_\mathcal{J} = \frac{1}{2}D(Y \parallel M) + D(S \parallel M), \tag{8.15}$$

$$\mathcal{D}(\mathcal{P} \parallel \mathcal{Q}) = \sum_{x \in X} P(x) \log(\frac{P(x)}{Q(x)}) \tag{8.16}$$

where $M = \frac{1}{2}(Y + S)$. Using $\mathcal{L}_\mathcal{J}$ instead of $\mathcal{L}_\mathcal{T}$ results in an MoF of 25.3%, which shows the superiority of our self supervision loss.

8.5.2.3 Effect of using different features

As previously stated, we leverage I3D features (*Carreira and Zisserman*, 2017) as input for our model. To assess the impact of the input video features, we also train our model using IDT, XViT(16$\times$) (*Bulat et al.*, 2021), and XViT+ATS(16$\times$) (*Fayyaz et al.*, 2022) features and present the results in Table 8.3. To provide a fair comparison with the method of *Richard et al.* (2018a), we also train this method using I3D, XViT(16$\times$), and XViT+ATS(16$\times$) features utilizing the publicly available code. Our results demonstrate that our method achieves superior performance by utilizing

Method	XViT	XViT+ATS	I3D	IDT
ActionSet *Richard et al.* (2018a)	19.2*	19.0*	20.1*	23.3
Ours	27.9	27.4	**30.4**	26.6

Table 8.3: A comparison of our method with that of *Richard et al.* (2018a) for various features is presented. The experiments were conducted on the Breakfast dataset, and the MoF metric was used to evaluate performance. Our method demonstrates superior results using both types of features. *The source code of the paper has been used for this experiment.

Dataset	Break Fast	Holl. Ext.
	MoF	*jacc. idx*
ActionSet-monte-carlo *Richard et al.* (2018a)	23.3	9.3
ActionSet-text-based *Richard et al.* (2018a)	23.2	9.2
SCV *Li and Todorovic* (2020)	30.2	17.7
SCT (Ours)	30.4	17.7
POC *Lu and Elhamifar* (2022)	**42.4**	**33.5**

Table 8.4: Comparison of our method to state-of-the-art methods for weakly supervised temporal segmentation. It is crucial to emphasize that the research presented in this chapter has been published in CVPR 2020 (*Fayyaz and Gall*, 2020). To provide a thorough overview and foster a deeper understanding of the prevailing trends in the field, we have integrated contemporary state-of-the-art methods published at or subsequent to the publication date of this chapter

both types of features. The ActionSet method (*Richard et al.*, 2018a) does not perform well on I3D features, which may be attributed to limitations in its temporal architecture design that is not capable of effectively learning a temporal embedding from I3D features. We also observe that our method performs better using I3D features compared to XViT(16$\times$) and XViT+ATS(16$\times$). This superior performance may be attributed to the multimodal nature of I3D features, which are extracted by a model that incorporates both optical flows and RGB frames. In long untrimmed instructional videos, such as those found in the BreakFast dataset, motion modeling is crucial for understanding actions. In these videos, the scene, actors, and objects are typically consistent and do not change much over time. In contrast, hand and object movements are often the discriminative factors of actions. Thus, we posit that incorporating optical flows as input for the feature extraction model enables the representation of more discriminative features, resulting in improved temporal action segmentation performance.

8.5.3 Comparison to Other methods

The task of learning temporal action segmentation using action sets as weak supervision had been addressed only by *Richard et al.* (2018a) to the publication date of this chapter (*Fayyaz and Gall*, 2020). We compare our approach to this method and the concurrent works on two datasets. As it can be seen in Table 8.4, our method achieves superior results on all datasets compared to its preceding and concurrent methods. Figure 8.3 shows a qualitative result of our method for a video from the Breakfast dataset.

8.6 Summary

In this chapter, we presented a network specifically designed for temporal action segmentation. The network is trained on extended videos, which are solely annotated with the set of present actions. During the training process, the videos are divided into temporal regions containing only one action class, adhering to the set of annotated actions. We extensively evaluated the approach on two datasets, and for each dataset, the proposed network surpasses the performance of prior work. This summary highlights the key contributions and findings of the chapter, demonstrating the effectiveness of the proposed method in the domain of temporal action segmentation.

CHAPTER 9

Weakly Supervised Temporal Action Segmentation from Action Transcripts

This chapter is based on the following publication:
Fast Weakly Supervised Action Segmentation Using Mutual Consistency
Yaser Souri*, Mohsen Fayyaz*, Luca Minicullo, Gianpiero Francesca, Juergen Gall
IEEE Transactions on Pattern Analysis and Machine Intelligence, 2022.
* *denotes equal contribution.*

Below we will summarize the contributions of each author.

- **Yaser Souri and Mohsen Fayyaz**
This work is the result of a collaborative effort between Yaser Souri and Mohsen Fayyaz. Yaser Souri proposed the two-branch network architecture and the consistency loss, while Mohsen Fayyaz formulated the final version of the MuCon loss, which exhibited superior performance compared to other possibilities. Both Yaser Souri and Mohsen Fayyaz contributed significantly to the implementation and experimentation of the weakly supervised setting on the Breakfast dataset. Mohsen Fayyaz was responsible for obtaining the results on the Hollywood dataset, while Yaser Souri carried out the fully supervised and mixed supervised implementation and experimentation. Yaser Souri also conducted the reproducibility studies of previous works and produced the majority of the qualitative figures and visualizations. Both Yaser Souri and Mohsen Fayyaz made substantial contributions to the writing of this publication.

- **Luca Minciullo and Gianpiero Francesca**
This study was the result of a collaborative effort between the authors, characterized by supervision, discussions, and writing. Additionally, Gianpiero Francesca played a crucial role by co-funding the project and providing the computational resources utilized in this research.

- **Juergen Gall**
Juergen Gall supervised the project, and his contributions extended to discussion, writing, funding, and provision of the computational resources utilized in this study.

Contents

In the preceding chapter, we introduced a weakly supervised temporal action segmentation approach that leverages merely a set of actions for training purposes. While employing a set of actions for model training is cost-effective, using it as adequate supervision for temporal action segmentation models presents a considerable challenge. Considering the constraints of techniques utilizing weak action set supervision, it is crucial to explore methods that can capitalize on more robust forms of supervision. Transcripts serve as one such example of weak supervision, providing ordered lists of actions that denote the sequence in which actions transpire within a training video, albeit without specifying their temporal occurrence. As a final contribution, in this chapter, we present a novel end-to-end framework for weakly supervised action segmentation using a two-branch neural network. The network's dual branches independently predict redundant yet distinct action segmentation representations, and we incorporate a mutual consistency loss term to enforce consistency between these redundant representations. Our approach attains the accuracy of state-of-the-art methods while demonstrating marked improvements in efficiency.

9.1 Introduction

As previously discussed, the process of annotating precise temporal boundaries of actions in extensive videos can be both time-consuming and expensive. Therefore, action segmentation approaches that can leverage weaker forms of supervision are of considerable importance. One such approach involves the use of ordered lists of actions, referred to as "transcripts." These transcripts, which include sequences such as *spoon powder - pour milk - stir milk*, provide information about the order in which actions occur within a training video, but do not specify their exact timing. Consequently, this type of weak supervision has gained popularity in the field of computer vision (*Huang et al.*, 2016; *Richard et al.*, 2017, 2018b; *Ding and Xu*, 2018; *Chang et al.*, 2019; *Li et al.*, 2019).

To learn from transcripts, previous approaches try to align the transcripts to the training videos, *i.e.*, they infer frame-wise labels of each training video based on the provided transcripts. This alignment is then used as the pseudo ground truth for training. For the transcript alignment, the Viterbi algorithm is commonly used. It takes the estimated frame-wise class probabilities of a video and

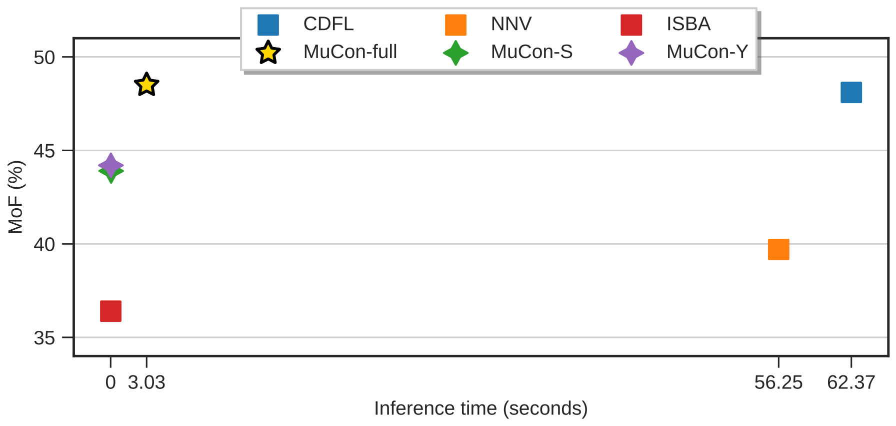


Figure 9.1: Average inference time per video (seconds) vs. average mean over frames (MoF) accuracy (%) for weakly supervised action segmentation approaches on the Breakfast dataset (*Kuehne et al.*, 2014). The average MoF is calculated over 5 training/inference iterations. The proposed approach (MuCon-full) provides the best trade-off between inference time and accuracy. More details are provided in Section 9.4.4.

finds the best sequence of frame labels that does not violate the action order of the given transcript. While *Richard et al.* (2017) and *Ding and Xu* (2018) perform the alignment after each epoch for all training videos, *Richard et al.* (2018b) and *Li et al.* (2019) apply it at each iteration to a single video.

During inference, previous approaches except ISBA (*Ding and Xu*, 2018) rely on segmentation through alignment. This means that given an unseen video from the test set, the methods search over all transcripts of the training set and take the transcript that best aligns with the test video. This is highly undesirable since the inference time increases in this case as the number of different transcripts in the training set increases. As a result, these approaches are inefficient as shown in Figure 9.1.

In contrast to segmentation through alignment approaches, ISBA is fast, but it does not achieve the accuracy of state-of-the-art approaches. This means that at the moment one can only choose between accurate or fast approaches, but one cannot have both.

In this work, we therefore fill this gap and propose an approach that is nearly as fast as ISBA and nearly as accurate as the state-of-the-art as shown in Figure 9.1. Instead of optimizing over all possible transcripts during inference, our approach directly predicts the transcript for a video as well as the frame-wise class probabilities using two branches as illustrated in Figure 9.2. This means that the inference time does not depend on the number of transcripts used for training and that the best label sequence can be directly predicted from the estimated transcript and frame-wise class probabilities.

In addition, the proposed approach has also two major advantages during training. First, the branch, which predicts the transcripts, is directly trained with the type of supervision that is provided, namely transcripts. Second, the branch not only predicts the transcripts, but also the length of each action in the transcript. This means that the branch predicts already a full segmentation of the video without the need for the Viterbi algorithm, which also reduces the training time. Nevertheless, we need an additional loss to learn the action lengths. In contrast to previous works, we do not generate pseudo ground truth with hard labels during training, but we propose a novel differentiable mutual consistency (MuCon) loss that enforces that the representations estimated by the two branches are mutually consistent and match each other. Furthermore, we show that the approach can be trained using weak supervision, full supervision, or a mixture of the two, where only few videos are fully annotated.

We provide a thorough analysis of the proposed approach including a detailed statistical analysis. We show that the proposed network with the mutual consistency loss achieves an accuracy that is either on par or better than existing approaches. At the same time, it is 20 times faster during inference compared to the most accurate approach by *Li et al.* (2019) as shown in Figure 9.1.

9.2 Weakly Supervised Action Segmentation

As mentioned in the previous section, action segmentation is the task of predicting the action class for each frame of a video. More formally, given an input sequence of T D-dimensional frame-level features $X_{1:T} = (x_1, \ldots, x_T)$, $x_t \in \mathbb{R}^D$, the goal is to predict the output sequence of frame-level action labels $\hat{Y}_{1:T} = (\hat{y}_1, \ldots, \hat{y}_T)$, where $\hat{y}_t \in \mathcal{C}$ and $\mathcal{C}$ is the set of action classes. The frame-level action labels $\hat{Y}_{1:T}$ can also be represented as an ordered sequence of M segments $\hat{S}_{1:M}$ where each segment $\hat{s}_m$ is defined as an action label $\hat{a}_m \in \mathcal{C}$ and its corresponding length $\hat{\ell}_m \in \mathbb{R}_+$.

For fully supervised action segmentation, the target labels for every frame $\hat{y}_t$ are known during training. This means that the target lengths $\hat{\ell}_m$ are known. However, in weakly supervised action segmentation, the only supervisory signal is the ordered sequence of actions $\hat{A}_{1:M} = [\hat{a}_1, \ldots, \hat{a}_m]$, often called *video transcript*, while the action lengths $\hat{L}_{1:M} = [\hat{\ell}_1, \ldots, \hat{\ell}_M]$ are unknown.

In order to estimate $L_{1:M}$, we exploit the fact that the two unknown target representations $\hat{Y}_{1:T}$ and $\hat{S}_{1:M}$ for action segmentation are redundant and it is possible to generate one given the other. The main idea is therefore to predict both representations $Y_{1:T}$ and $S_{1:M}$ by different branches of the network such that they can supervise each other. In the ideal case, both predicted representations converge to the same solution during training. In order to train the model, however, we have to face the challenging problem that the mapping from one representation to the other needs to be differentiable, such that the loss that measures the mutual consistency of both representations is differentiable with respect to the predicted frame-wise class probabilities $Y_{1:T}$ and the predicted segment lengths $L_{1:M}$. Since the transcript $\hat{A}_{1:M}$ is given, it does not need to be differentiable with respect to $A_{1:M}$.

9.3 Proposed Method

During training, we have for each video the input video features $X_{1:T}$ and its corresponding transcript $\hat{A}_{1:M}$. Since only weak labels in form of the transcripts are provided for training, we propose a) to use the weak labels directly for training a transcript prediction subnetwork and b) exploit the two representations discussed in Section 9.2 as mutual supervision.

The proposed network is illustrated in Figure 9.2. The network consists of two branches that share the same backbone $f_t(X)$, which is a temporal convolutional network that maps the input video features $X \in \mathbb{R}^{T \times D}$ to a latent video representation $Z \in \mathbb{R}^{T' \times D'}$. The backbone architecture is described in Section 9.3.1. After the shared backbone, the network has two branches. The top branch $f_c(Z)$ is a standard frame classification branch that estimates the class probabilities of each frame $Y \in \mathbb{R}^{T \times N}$, where N is the number of classes. The branch will be described in Section 9.3.2.

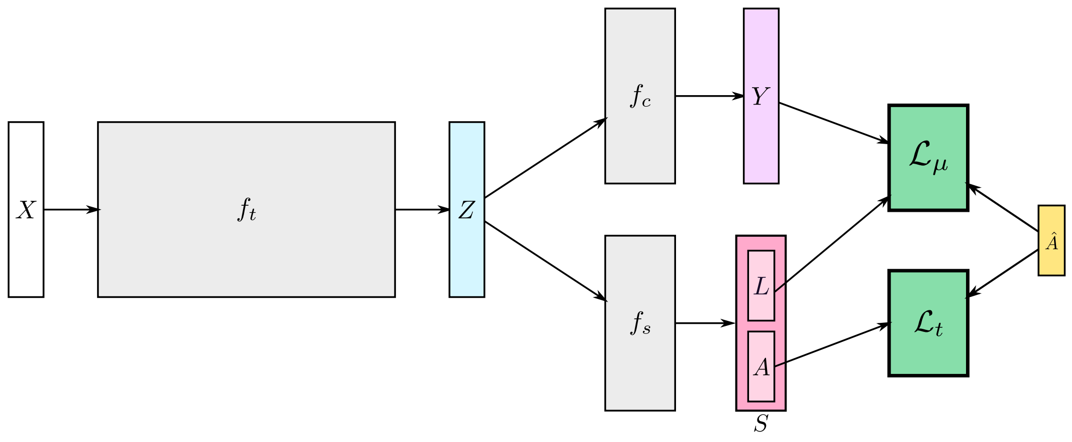


Figure 9.2: Our proposed network consists of three subnetworks (gray). The temporal backbone f_t embeds the input features in the hidden representation Z which is used for two branches. While the frame classification branch f_c predicts framewise class probabilities Y for action segmentation, the segment generation branch f_s predicts the segment representation S for action segmentation. We train our network using two loss functions. While the transcript prediction loss $\mathcal{L}_t$ enforces that the predicted transcript A matches the ground-truth transcript $\hat{A}$, our proposed mutual consistency (MuCon) loss $\mathcal{L}_\mu$ enforces that the two representations are consistent.

The novelty of the network is the lower branch and the mutual consistency loss $\mathcal{L}_\mu$, which will be described in Sections 9.3.3 and 9.3.4, respectively.

In contrast to previous works that commonly use a network to predict Y and Viterbi decoding on top of it for computing the loss based on the given transcript $\hat{A}$, we use a second branch that predicts the segment representation S, *i.e.* A and L. This has two advantages during training. First, we can compare the predicted transcript A and the ground-truth transcript $\hat{A}$ for each video directly, *i.e.*, the network is directly trained with the type of supervision that is provided. The corresponding loss is denoted by $\mathcal{L}_t$. Second, the network has two branches that supervise each other and we do not need an additional Viterbi decoding step. Since the two branches predict different representations, the novel mutual consistency (MuCon) loss $\mathcal{L}_\mu$ requires a differentiable mask generator such that the loss is differentiable with respect to Y and L; otherwise we could not train the network.

After training, we need to infer the action class for each frame of an unseen test video. In case of action segmentation, no additional transcripts are given for the test videos. As shown in Figure 9.1, we can use three different approaches for inference, which are denoted by MuCon-Y, MuCon-S, and MuCon-full. In case of MuCon-Y and MuCon-S, we use the predicted Y or S representation, respectively. While these settings are as fast but more accurate than ISBA (*Ding and Xu*, 2018), the accuracy is lower compared to the state-of-the-art. For the model MuCon-full, we use therefore the predictions of both branches. To this end, we use the predicted framewise class probabilities Y and the predicted transcript A to find the best sequence of action labels using Viterbi decoding as described in Section 9.3.8. Note that the approach is still much faster than other methods except of ISBA since we do not need to optimize over all possible transcripts. We now provide more details for each part of the network.

9.3.1 Temporal Backbone

The temporal backbone $f_t(X)$, which is a temporal convolutional network (*van den Oord et al.*, 2016; *Fayyaz and Gall*, 2020; *Abu Farha and Gall*, 2019; *Lea et al.*, 2017; *Li et al.*, 2020a), outputs the latent video representation $Z \in \mathbb{R}^{T' \times D'}$ of the input video features $X \in \mathbb{R}^{T \times D}$. This hidden representation has a smaller temporal resolution due to temporal pooling. Similar to the chapter 8 our temporal backbone consists of a set of 1-dimensional dilated convolutional layers with increasing dilation sizes. More specifically, we first apply a 1-d convolution with kernel size 1 to perform dimensionality reduction. Then a set of 11 layers with increasing dilation rates are applied, followed by a single 1-d convolution with kernel size 1 that generates the output. The amount of dilation for each layer is 2^i. We perform temporal max pooling with a kernel size of 2 after the layers 1, 2, 4, and 8. Additional details are given in Section 9.4.2. The design is similar to a single stage TCN (*Abu Farha and Gall*, 2019), but it includes additional pooling layers.

9.3.2 Frame Classification Branch

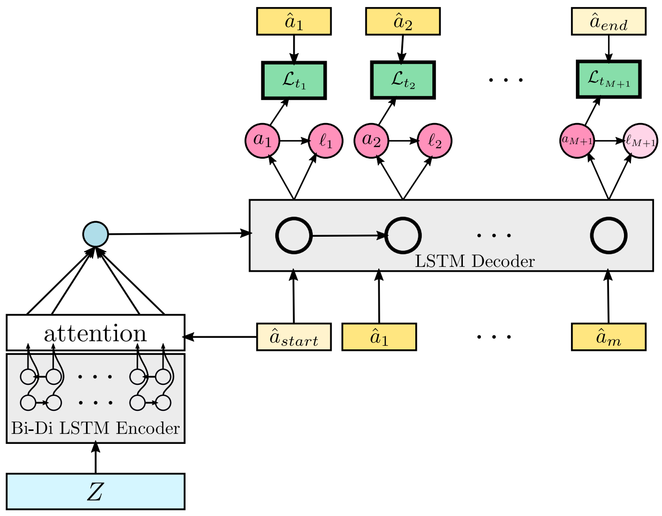
The classification branch takes the shared latent video representation $Z \in \mathbb{R}^{T' \times D'}$ as input and predicts the class probabilities $Y \in \mathbb{R}^{T \times N}$ where N is the number of actions in the dataset. Due to the temporal pooling of the temporal backbone, Z has a lower temporal resolution compared to the input features. To compensate for this, we first upsample Z using nearest-neighbor interpolation to the desired temporal size T. Then we use a single 1-d convolution with kernel size 1, which takes as input the upsampled $Z \in \mathbb{R}^{T \times D'}$ and outputs $Y \in \mathbb{R}^{T \times N}$.

9.3.3 Segment Generation Branch

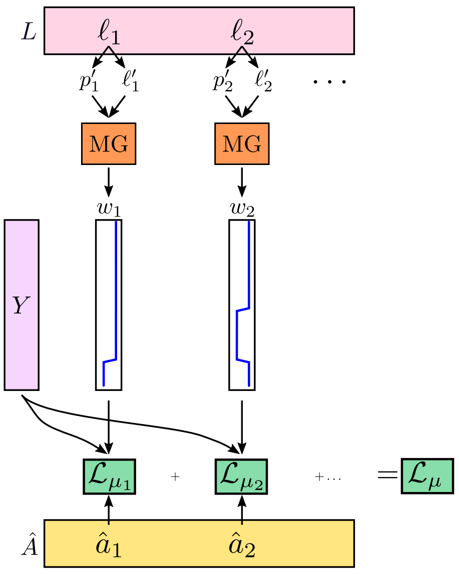
The second branch on top of Z is the segment generation subnetwork which predicts the segments S. Each segment s_m consists of predicted action probabilities a_m and the estimated relative log length ℓ_m of that segment. By predicting the log length, we implicitly enforce that the segment lengths are positive. We will discuss in Section 9.3.5.1 how the relative log length is mapped to the absolute length. The subnetwork and the transcript prediction loss $\mathcal{L}_t$ are illustrated in Figure 9.3a.

We employ a conventional sequence to sequence network with attention (*Bahdanau et al.*, 2015). Given the hidden video representation Z, we use a bidirectional LSTM encoder to encode it. Our decoder is an LSTM recurrent neural network with MLP attention. Although these networks on their own struggle to learn a temporal model for long input sequences (*Cho et al.*, 2014b; *Singh et al.*, 2016), the temporal backbone, which encodes temporal relations at higher resolution, makes it easier for the segment generation subnetwork to learn temporal dependencies as shown by our ablation experiments.

As illustrated in Figure 9.3a, the decoding starts with the starting symbol $\hat{a}_{start}$. At each step of the decoding process, the ground truth action label $\hat{a}_{m-1}$ from the previous step is added to the encoded input sequence by concatenating it to the result of the attention. The concatenated vector is then given as input to the LSTM decoder cell, which estimates probability scores a_m using a fully connected MLP with two layers. Given these probability scores and the ground truth action label $\hat{a}_m$, we compute the action prediction loss $\mathcal{L}_{t_m}$ per segment using cross-entropy. The final transcript prediction loss is defined as the sum of the action prediction losses, *i.e.*, $\mathcal{L}_t = \sum_{m=1}^{M+1} \mathcal{L}_{t_m}$. Note that we have $M + 1$ terms since we add the end symbol $\hat{a}_{end}$ to the ground-truth transcripts, which



(a)



(b)

Figure 9.3: Visualization of the segment generation branch and the loss functions. **(a)** Segment generation branch f_s and the transcript prediction loss $\mathcal{L}_t$. Given the hidden video representation Z, we use a sequence to sequence network with attention. The transcript prediction loss $\mathcal{L}_t$ compares the predicted action class probabilities a_m with the ground-truth action label $\hat{a}_m$. **(b)** Mutual consistency loss $\mathcal{L}_\mu$. Given the predicted lengths L, a set of masks w_m are generated using differentiable sampling by the mask generation module (MG). The loss then measures for each segment the consistency of the estimated framewise class probabilities Y with the ground-truth action $\hat{a}_m$.

needs to be predicted by the network as well. During training we use teacher forcing **?**, *i.e.*, we do not sample from the predicted action probabilities to feed the decoder, but we use the ground truth action labels $\hat{a}_m$.

Given the probability scores a_m and the hidden state of the decoder, we use another fully connected MLP with two layers to predict ℓ_m, which corresponds to the logarithm of the relative length of the segment. Notice that the parameters of this MLP are not updated based on the transcript prediction loss, but based on the mutual consistency loss $\mathcal{L}_\mu$.

9.3.4 Mutual Consistency Loss

Using the frame classification and the segment generation branches, two representations Y and S for the action segmentation are produced. However, so far we have only defined a loss for the predicted transcript A of the segment representation S, which we compare directly with the ground-truth transcript $\hat{A}$, but not for the predicted lengths of the actions L and the framewise class probabilities Y. We therefore propose the mutual consistency (MuCon) loss, which enforces that the two representations match each other and are mutually consistent. As shown in Figure 9.2, the mutual consistency loss takes the ground-truth transcript $\hat{A}$, the predicted segment lengths L, and framewise class probabilities Y as input. Since the loss is used to train both branches, it needs to be differentiable with respect to L and Y.

In principle, there are two choices: either we map Y to S or S to Y. The first approach, however, is not practical since it would need to detect consistent segments within Y. While this could be done using Viterbi decoding, it would make our network as expensive as segmentation through alignment approaches. Our goal, however, is to propose an approach that is fast and accurate. An explicit mapping from S to Y followed by the computation of a framewise loss, however, is also inefficient. We therefore combine the mapping and loss computation. Furthermore, we use $\hat{A}$ instead of A since we have already a loss that enforces that A is close to $\hat{A}$. Like teacher forcing, it also stabilizes the training since it guides the mutual consistency loss by $\hat{A}$ from the beginning of the training when A is still noisy. In our experiments, we show that this reduces the standard deviation over different runs.

The computation of the mutual consistency loss is illustrated in Figure 9.3b. We start with the estimated relative log length ℓ_m for each segment s_m. The relative log length ℓ_m is then converted into the absolute length ℓ'_m and for each segment we compute its absolute starting position p'_m within the video. This step is described in Section 9.3.5.1. Given ℓ'_m and p'_m, for each segment we generate a mask w_m using the differentiable mask generation module, which is described in Section 9.3.5. As mentioned before, for each segment we use the ground-truth action label $\hat{a}_m$ instead of the estimated class probabilities.

Finally, we can compare the consistency of the predicted frame-wise class probabilities Y with each segment defined by the mask w_m and the label $\hat{a}_m$. To this end, we first compute the average of Y for each segment based on the mask w_m:

$$g(Y, w_m) = \frac{\sum_{t=1}^{T} y_t w_m[t]}{\ell'_m} \tag{9.1}$$

where $g(Y, w_m) \in \mathbb{R}^N$, $w_m[t]$ is the value of the mask at frame t, and ℓ'_m is the absolute length of

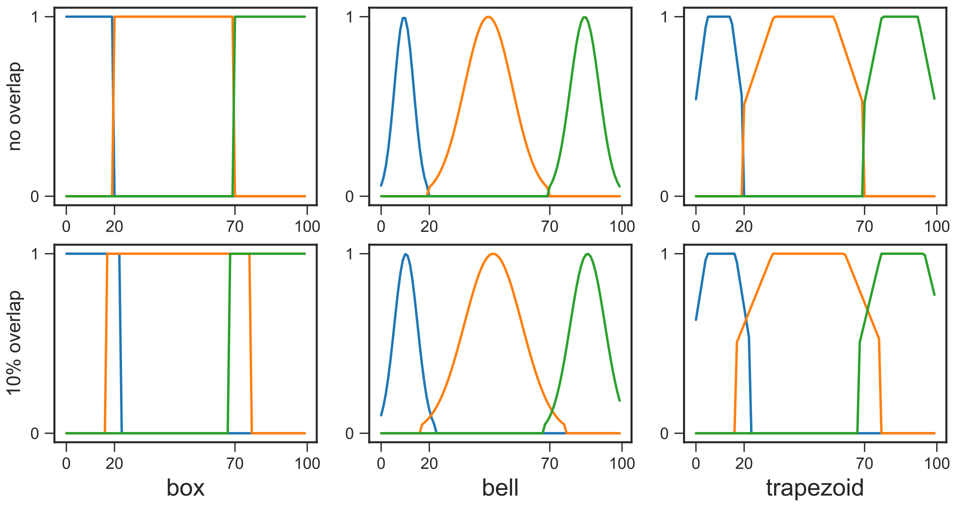


Figure 9.4: Examples of different masks for three consecutive action segments. The top row shows regular masks with different shapes while the bottom row shows masks generated with added 10% overlap. The left, middle and right figures depict box, bell, and trapezoid shaped masks.

the segment. Then, we compute the cross-entropy for each segment:

$$\mathcal{L}_{\mu_m}(Y, w_m, \hat{a}_m) = -\log\left(\frac{e^{g(Y,w_m)[\hat{a}_m]}}{\sum_{n=1}^{N} e^{g(Y,w_m)[n]}}\right) \tag{9.2}$$

where we use the softmax function to normalize the class probabilities per segment and the ground-truth label $\hat{a}_m$. Since we normalize per segment using the softmax function, we use the estimates of Y before the softmax layer of the frame classification branch in (9.1). The final mutual consistency loss $\mathcal{L}_\mu$ is defined as the sum of all segment losses:

$$\mathcal{L}_\mu = \sum_{m=1}^{M} \mathcal{L}_{\mu_m}(Y, w_m, \hat{a}_m). \tag{9.3}$$

The mutual consistency loss (9.3) is a differentiable function of the masks w_m and the frame-wise class probabilities Y. Due to the definition of the differentiable mask generation, which we will describe in the following section, the masks are differentiable with respect to the estimated segment lengths L. This means that the gradients of the mutual consistency loss are backpropagated through both branches as it is required to train the network.

9.3.5 Differentiable Mask Generation

In order to compute the mutual consistency loss, we need to generate the masks w_m that act like gating functions that only allow information from the predicted temporal regions to pass through. The masks w_m are functions of the predicted absolute length ℓ'_m and predicted starting position p'_m of each segment such that

$$w_m[t] \simeq \begin{cases} 1 & p'_m \leqslant t \leqslant p'_m + \ell'_m \\ 0 & otherwise \end{cases}, \quad t \in [1 \dots T]. \tag{9.4}$$

Examples of generated masks are shown in Figure 9.4.

9.3.5.1 Localization

In order to obtain the predicted absolute length ℓ'_m and the predicted starting position p'_m for each segment, we first calculate the absolute length values $L'_{1:M} = (\ell'_1, \ldots, \ell'_M)$ for a video with T frames such that $\sum_{m=1}^{M} \ell'_m = T$, *i.e.*, the absolute lengths sum up to be equal to the length of the video. Having the absolute length ℓ'_m of each segment, we can also compute the absolute starting position p'_m for each segment:

$$\ell'_m = T \frac{e^{\ell_m}}{\sum_{k=1}^{M} e^{\ell_k}}, \quad p'_1 = 0, \quad p'_m = \sum_{k=1}^{m-1} \ell'_k. \tag{9.5}$$

9.3.5.2 Temporal Transformation

For generating the masks, we transform a reference template tensor $U \in [0,1]^J$ to $w_m \in [0,1]^T$ where J is a canonical value equal to 100. The reference template tensor can be of any shape and we evaluate three shapes, namely *box*, *bell*, and *trapezoid* which are depicted in Figure 9.4. By adjusting the absolute lengths ℓ'_m and starting positions p'_m, it is also possible to introduce an overlap for the masks.

We transform U to w_m by scaling and translating it using ℓ'_m and p'_m, respectively. Therefore, we use a 1D affine transformation matrix $\mathcal{T}_\theta$ such that

$$i_u[t] = \mathcal{T}_\theta(t) = [\theta_0 \ \theta_1] \begin{pmatrix} t \\ 1 \end{pmatrix} \quad \forall t \in [1, \ldots, T] \tag{9.6}$$

where $i_u[t]$ are the sampling points over U. As a result of the affine transformation (9.6), the element indices of the sampling points in the template $i_u[t]$ can be outside the valid range of $[1, \ldots, J]$. In such cases, the indices will be ignored in the sampling process (9.8). The affine transformation parameters are

$$\theta_0 = \frac{J}{\ell'_m}, \quad \theta_1 = \frac{-Jp'_m}{\ell'_m} \tag{9.7}$$

where θ_0 scales the reference template U to the estimated length ℓ'_m and θ_1 translates it to the estimated position p'_m.

9.3.5.3 Temporal Sampling

To perform the aforementioned temporal transformation, we should sample from U using the sampling points $i_u[t]$ and produce the sampled mask w_m. Each index $i_u[t]$ refers to the element in U where a sampling kernel must be applied to get the value at the corresponding element $w_m[t]$ in the output mask. Similar to Jaderberg *et al.* (2015), we perform this operation as follows

$$w_m[t] = \sum_{j=1}^{J} U[j] \Psi(i_u[t] - j) \quad \forall t \in [1, \ldots, T] \tag{9.8}$$

where Ψ is the sampling kernel. Since we use a linear kernel, it can be written as

$$w_m[t] = \sum_{j=1}^{J} U[j] \max(0, 1 - |i_u[t] - j|) \quad \forall t \in [1, \dots, T]. \tag{9.9}$$

9.3.5.4 Backpropagation

To be able to backpropagate through the generated masks w_m, we define the gradients with respect to the sampling indices $i_u[t]$ as

$$\frac{\partial w_m[t]}{\partial i_u[t]} = \sum_{j=1}^{J} U_j \begin{cases} 0 & |i_u[t] - j| \geqslant 1 \\ 1 & j - 1 < i_u[t] \leqslant j \\ -1 & j < i_u[t] < j + 1 \end{cases} . \tag{9.10}$$

Since the sampling indices $i_u[t]$ are a function of the predicted lengths ℓ'_m, the loss gradients are backpropagated to the segment generation branch.

9.3.6 Regularization

To prevent degenerate solutions, *i.e.*, solutions where the lengths of some segments are large and the length of the other segments are almost zero, we add a regularization term for the predicted relative log lengths L:

$$\mathcal{L}_\ell = \sum_{m=1}^{M} \max(0, -\ell_m - w) + \max(0, \ell_m - w). \tag{9.11}$$

The proposed length regularizer adds a penalty if $\ell_m < -w$ or $\ell_m > w$. Note that the relative log lengths can be negative and are later converted into the absolute length as described in Section 9.3.5.1.

We also use the smoothing loss introduced by *Abu Farha and Gall* (2019) for the frame classification branch:

$$\mathcal{L}_s = \frac{1}{TN} \sum_{t,n} \hat{\Delta}_{t,n}^2, \tag{9.12}$$

$$\hat{\Delta}_{t,n} = \min(\tau, |\log y_{t,n} - \log y_{t-1,n}|), \tag{9.13}$$

with $\tau = 4$ following *Abu Farha and Gall* (2019).

9.3.7 Fully Supervised and Mixed Training

Although our approach is designed for weakly supervised action segmentation, we can easily adapt it to a setting where all or some videos of the training set are fully annotated, *i.e.*, they are annotated not by the transcript $\hat{A}$ but by the framewise labels $\hat{Y}$.

In this case, we have two additional losses for our network shown in Figure 9.2. We use the cross-entropy loss for the frame classification branch to compare the predicted class probabilities Y with the framewise labels $\hat{Y}$, and we use the mean squared error loss for the segment generation branch to compare the estimated relative log lengths L with the ground truth segment lengths. To this end, we convert the absolute ground-truth lengths into relative log lengths.

Although the mutual consistency loss is only necessary in case of weakly supervised learning or in a setting where only a subset of the videos is fully annotated, we show that using the mutual consistency loss also improves the accuracy when the network is trained in a fully supervised way.

9.3.8 Inference

As discussed at the beginning of Section 9.3, we can either use the frame classification branch, which predicts the framewise class probabilities Y, or the segment generation branch, which predicts the segments S. In the later case, we start with the start symbol a_{start} and the decoder generates new segments until the special end symbol a_{end} is predicted as illustrated in Figure 9.3a. The estimated relative log segment lengths are then converted into absolute lengths as described in Section 9.3.5.1. The two approaches for inference are denoted by MuCon-Y and MuCon-S, respectively.

However, as it is shown in Figure 9.1, we achieve the highest accuracy at a small increase in inference time if we use the predictions of both branches. We denote this approach by MuCon-full. For simplicity, we use L instead of L' to denote the estimated absolute lengths of the segments in this section. In order to fuse both predictions, we keep the inferred transcript A, but reestimate the lengths of the segments L using Y:

$$L^* = \underset{\tilde{L}}{\operatorname{argmax}}\, p(\tilde{L}|Y, A, L). \tag{9.14}$$

Similar to *Richard et al.* (2018b), we can factorize the term $p(\tilde{L}|Y, A, L)$ yielding

$$L^* = \underset{\tilde{L}}{\operatorname{argmax}} \prod_{t=1}^{T} y_t[a_{\alpha(t,\tilde{L})}] \prod_{m=1}^{M} P_{\ell_m}(\tilde{\ell}_m) \tag{9.15}$$

where $P_{\ell_m}(\tilde{\ell}_m)$ denotes a Poisson distribution with expected mean ℓ_m, which corresponds to the absolute segment length that has been estimated by the segment generation branch. Depending on $\tilde{L}$, the segment number changes for a frame t and it is denoted by $\alpha(t, L)$. Although L^* is obtained by dynamic programming as described by *Richard et al.* (2018b), we do not need to optimize over all possible transcripts as in the methods of *Li et al.* (2019) and *Richard et al.* (2018b). While *Li et al.* (2019) and *Richard et al.* (2018b) align each transcript of the training set to the test video and take the training transcript that best aligns to the test video, our approach infers the transcript A directly from the test video and only aligns A to the test video. MuCon-full is therefore still much faster than the methods of *Li et al.* (2019) and *Richard et al.* (2018b) as shown in Figure 9.1.

9.4 Experiments

We evaluate our approach for two tasks, namely action segmentation as described in Section 9.2 and action alignment. In contrast to action segmentation, the transcripts are also given for the test videos in case of action alignment. Besides of the weakly supervised setting, we also evaluate the approach when it is trained fully supervised or in a mixed setting where some videos are fully annotated and the other videos are only weakly annotated by transcripts. Before we evaluate the approach, we discuss the evaluation protocols and further implementation details[1].

[1]Source code is available at: github.com/MohsenFayyaz89/MuCon

9.4.1 Evaluation Protocols and Datasets

We evaluate our method on two popular datasets, the Breakfast dataset (*Kuehne et al.*, 2014) and the Hollywood extended dataset (*Bojanowski et al.*, 2014). The Breakfast dataset contains more than 1.7k videos of different cooking activities. The videos contain 10 different types of breakfast activities such as *prepare cereal* or *prepare coffee* which consists of 48 different fine-grained actions. In our experiments, we follow the 4 train/test splits provided with the dataset and report the average. The Hollywood extended dataset contains 937 video sequences taken from Hollywood movies. The videos contain 16 different action classes. We follow the train/test split strategy of *Richard et al.* (2017), *Richard et al.* (2018b), and *Li et al.* (2019).

The main performance metric used for the action segmentation task is the mean over frames (MoF) accuracy (*Abu Farha and Gall*, 2019; *Kuehne et al.*, 2016b; *Richard et al.*, 2017, 2018b). We also report the F1 score that has been used by fully supervised approaches (*Abu Farha and Gall*, 2019; *Li et al.*, 2020a). For action segmentation, we also directly evaluate the performance of the predicted transcripts, which is measured by the matching score or edit distance (*Lambert*, 2019). For action alignment, we use the intersection over detection (IoD) metric following *Bojanowski et al.* (2014), *Richard et al.* (2017), *Richard et al.* (2018b), *Ding and Xu* (2018), *Chang et al.* (2019), and *Li et al.* (2019).

9.4.2 Implementation Details

We train the entire network end-to-end after initializing it with Gaussian random weights. In each iteration, we use only a single video, *i.e.*, the batch size is 1. We use a single group norm (*Wu and He*, 2018) layer with 32 groups after the final layer of the temporal backbone f_t. The dimensionality of the shared latent video representation Z is $D' = 128$. Unless otherwise stated, we apply temporal max pooling with kernel size 2 after the convolutional layers 1, 2, 4, and 8 for all experiments on the Breakfast dataset. For the experiments on the Hollywood Extended dataset, we omit the last temporal max pooling after the convolutional layer 8 since the videos in the Hollywood extended dataset are relatively short. The size of the hidden states of the bidirectional LSTM encoder and the LSTM decoder in our segment generation module are set to 128. We also employ an input embedding for the LSTM decoder of size 128 with 0.25 dropout.

In the weakly supervised setting, the training loss is defined as $\mathcal{L} = \mathcal{L}_\mu + \mathcal{L}_t + \alpha\mathcal{L}_\ell + \beta\mathcal{L}_s$. We use $\alpha = \beta = 0.1$ unless otherwise specified. Since most of the frames in the Hollywood extended dataset are annotated by background, which increases the possibility of degenerate solutions, we set α to 1 for the Hollywood extended dataset. We use $w = 2$ in (9.11). The initial learning rate is set to 0.01 and is lowered by a factor of 10 after 70 epochs for the Breakfast dataset and after 60 epochs for the Hollywood extended dataset. We train our network for 150 epochs for all weakly supervised experiments. In the fully supervised and mixed supervision experiments, we train for 110 epochs since convergence is faster with full supervision.

As input features for the Breakfast and Hollywood extended datasets, we use RGB+flow I3D (*Carreira and Zisserman*, 2017) features extracted from a network that was pretrained on the Kinetics dataset (*Kay et al.*, 2017). These features have been previously used for fully supervised approaches (*Abu Farha and Gall*, 2019; *Li et al.*, 2020a). A recent study found that current weakly supervised approaches perform better with IDT features than I3D features (*Souri et al.*, 2020). We therefore report the results for the features where the corresponding methods perform best, *i.e.*, I3D

f_t	$\mathcal{L}_\mu$	Mat Score
single 1-d conv	✗	0.691
fixed dilation	✗	0.696
increasing dilation, linear	✗	0.727
increasing dilation, exponential	✗	0.729
increasing dilation, exponential	✓	0.774

Table 9.1: Transcript prediction accuracy on split 1 of the Breakfast dataset. The first four rows show the matching score of the segment generation branch for different dilation factors used for the temporal backbone. In these settings, the mutual consistency loss $\mathcal{L}_\mu$ is not used. The last row shows the proposed setting with the mutual consistency loss.

Shape	Overlap	MoF	Mat Score	F1@{10,25,50}		
box	✗	49.0	0.774	67.9	59.5	40.2
bell	✗	43.6	0.777	65.1	55.5	35.4
trapezoid	✗	48.7	0.773	67.3	59.0	39.4
box	10%	48.8	0.787	68.3	59.6	39.8
bell	10%	43.5	0.779	64.8	55.3	34.7
trapezoid	10%	48.0	0.780	67.7	59.1	39.5

Table 9.2: Impact of mask generation. As shown in Figure 9.4, different shapes with or without overlap can be used for the mask generation. We report the MoF accuracy, matching score, and F1 score.

for *Abu Farha and Gall* (2019) and *Li et al.* (2020a) and IDT for *Ding and Xu* (2018), *Richard et al.* (2018b), and *Li et al.* (2019).

9.4.3 Ablation Experiments

In this section, we quantitatively examine different components in our method. For each metric, We report the average and standard deviation over 5 runs on split 1 of the Breakfast dataset (*Kuehne et al.*, 2014).

9.4.3.1 Transcript Prediction

We first analyze how well the proposed approach predicts the transcripts, which is measured by the matching score. If we only use the segment generation branch, the network achieves a matching score of 0.729 as shown in row 4 of Table 9.1. If we add the frame classification branch and the mutual consistency loss $\mathcal{L}_\mu$, the matching score increases to 0.774. This shows that the mutual consistency loss also improves the transcript prediction.

In the first four rows of Table 9.1, we furthermore evaluate the impact of the dilation factors for the temporal backbone f_t. If we replace the temporal backbone by a single 1-d convolution, the matching score decreases from 0.729 to 0.691. If we use as in the proposed network 11 layers with 1-d convolutions but with fixed dilation factor of 1, the matching score does not significantly increase

Regularizer	MoF	Mat Score	F1@{10,25,50}		
none	47.4	0.787	68.1	59.3	39.7
$\mathcal{L}_s$	47.5	0.780	67.8	59.0	39.5
$\mathcal{L}_\ell$	48.3	0.777	67.8	59.0	40.3
$\mathcal{L}_\ell + \mathcal{L}_s$	49.0	0.774	67.9	59.5	40.2

Table 9.3: Impact of regularizers. The first three rows denote settings where the length regularizer $\mathcal{L}_\ell$ for the segment generation branch, the smoothing loss $\mathcal{L}_s$ for the frame classification branch, or both are omitted.

Inference variant	MoF	F1@{10,25,50}		
MuCon-Y	44.7	28.1	22.6	13.3
MuCon-S	43.6	65.6	57.2	33.8
MuCon-full	49.0	67.9	59.5	40.2

Table 9.4: Impact of fusing both branches. While MuCon-Y and MuCon-S use either the frame classification branch or the segment generation branch for inference, MuCon-full fuses both branches for inference.

since it does not substantially increase the receptive field. Only when we linearly or exponentially increase the dilation factors for each layer, we observe an improvement of the matching score.

9.4.3.2 Mask Generation Settings

As described in Section 9.3.5, different shapes of mask templates can be used for the mutual consistency loss. We evaluated three shapes, namely *box*, *bell*, and *trapezoid*, and added an optional overlap of 10% as depicted in Figure 9.4. The results reported in Table 9.2 show that the shape of the mask has little impact on the quality of the predicted transcript, but that the bell shape has the worst performance. In general, having hard boundaries for the masks without any overlap performs best. For all other experiments, we therefore use the *box* shape without any overlap.

9.4.3.3 Effect of Regularizers

As described in Section 9.3.6, we also use two additional regularizers during training, namely the length loss $\mathcal{L}_\ell$ and the smoothing loss $\mathcal{L}_s$. We evaluate the impact of each term in Table 9.3 by removing $\mathcal{L}_\ell$, $\mathcal{L}_s$, or both. Without any regularizer, the MoF accuracy decreases from 49.0 to 47.4 and F1@50 from 40.2 to 39.7. While adding only the smoothing loss does not result in an improvement, the length regularizer and the combination of both improves MoF.

9.4.3.4 Different Variants for Inference

As described in Section 9.3.8, we have three options for inference. We can use the frame classification branch, the segment generation branch, or both. The three approaches are denoted by MuCon-Y, MuCon-S, and MuCon-full, respectively. The results in Table 9.4 show that fusing the two branches improves the MoF accuracy compared to MuCon-S by more than 5% and F1@50 by more than 6% at

Teacher forcing	MoF	Mat Score	F1@{10,25,50}		
None	48.7	0.779	68.2	59.5	40.3
70 epochs	48.9	0.783	68.2	59.5	39.8
All epochs	49.0	0.774	67.9	59.5	40.2

Table 9.5: Impact of teacher forcing. The first row denotes a setting where teacher forcing is not used for training. The second row denotes a setting where teacher forcing is used only for the first 70 epochs. The last row denotes a setting where teacher forcing is used for all epochs.

Approach	Avg MoF	Avg Mat Score	Training (hours)	Inference (seconds)
ISBA *Ding and Xu* (2018)	36.4	-	12.75	0.01
NNV *Richard et al.* (2018b)	39.7	0.686	11.23	56.25
CDFL *Li et al.* (2019)	48.1	0.712	66.73	62.37
MuCon-Y	44.2	-	4.57	0.02
MuCon-S	43.9	**0.785**	4.57	0.04
MuCon-full	**48.5**	**0.785**	4.57	3.03

Table 9.6: Comparison of accuracy, training, and inference time. We report the average and standard deviation of MoF and matching score for 5 runs on the entire Breakfast dataset (*Kuehne et al.*, 2014). For ISBA and MuCon-Y, the matching score is not calculated as they predict frame-wise labels and not the transcript. Training and inference time is measured as wall clock time. The training time is measured for training on split 1 of the Breakfast dataset (*Kuehne et al.*, 2014). The inference time is measured as the average inference time for a video from the test set of split 1.

a small increase in inference time as shown in Figure 9.1. Since MuCon-Y predicts only frame-wise labels and not segments, the F1 scores are very low for MuCon-Y.

9.4.3.5 Effect of Teacher Forcing

As described in Sections 9.3.3 and 9.3.4, we use the ground truth transcript while predicting the transcript and calculating the MuCon loss in order to stabilize the training at the beginning. In Table 9.5, we evaluate the impact of teacher forcing where we also include a setting that uses teacher forcing only for the first 70 epochs of training. The results show that teacher forcing does not impact the average accuracy but it reduces the standard deviation, which is an indicator of a stable training procedure.

9.4.4 Training and Inference Time

Besides of the accuracy, we also compare the training and inference time of our approach with CDFL (*Li et al.*, 2019), NNV (*Richard et al.*, 2018b), and ISBA (*Ding and Xu*, 2018). For a fair comparison, we always used the same hardware. For training, we used a machine with an Nvidia GeForce GTX 1080Ti GPU, 188 GB of RAM, and an Intel(R) Xeon(R) Gold 5120 (2.20GHz) CPU. For testing, we used a machine with an Nvidia GeForce GTX Titan X GPU, 32 GB of RAM, and an Intel(R) Core(TM) i7-4930K (3.40GHz) CPU. We only calculate the wall time for training and deactivated for all methods any unnecessary operations like saving intermediate results. Since we

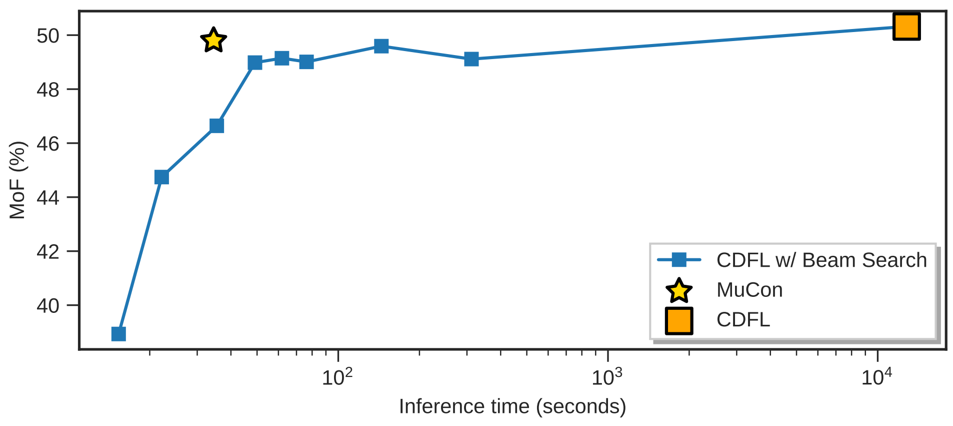


Figure 9.5: Using beam search to reduce the inference time of CDFL. The blue squares show the performance of CDFL with different values for the beam size. The x-axis is the total inference time for split 1 of the Breakfast dataset in log scale.

used pre-computed features for all experiments, the time measurement includes the time to load the features but not the time to compute the features. In average, computing the features takes 92 seconds per video for the Breakfast dataset where 72.5 seconds are spent for calculating the optical flow.

The results are reported in Table 9.6. We observe that our approach is 14 times faster to train and 20 times faster during inference compared to the state-of-the-art approach CDFL (*Li et al.*, 2019). As mentioned before, our approach does not perform any Viterbi decoding during training, which makes it faster during training. Also and most importantly, our approach only performs one Viterbi decoding step for the estimated transcript during inference as compared to CDFL and NNV which need to optimize over all possible transcripts. This shows that our approach offers by far the best trade-off between accuracy and runtime as it is also illustrated in Figure 9.1.

In order to avoid the full alignment of all transcripts of the training set to a test video, beam search can be used. Beam search limits the maximum number of hypotheses and removes hypotheses with low probability at an early stage. The smaller the beam size, i.e., the number of hypotheses, is, the faster is the inference. This, however, comes at the cost of reducing the accuracy. In order to evaluate how much runtime reduction can be achieved by beam search, we evaluate CDFL with beam search. Figure 9.5 shows the MoF accuracy and inference time for different beam sizes. In contrast to Figure 9.1, we report the accuracy for only one run and only split 1 of the Breakfast dataset. We furthermore plot the inference time of the entire test set in log scale for better visualization. We observe that beam search reduces the inference time of CDFL by multiple orders of magnitude but also its accuracy. When we compare a setting where CDFL with beam search is as fast as MuCon, we observe that MuCon achieves a much higher accuracy.

9.4.5 Further Comparisons

We therefore compare our method with state-of-the-art methods based on the reported numbers on the Breakfast dataset (*Kuehne et al.*, 2014) in Table 9.7 and on the Hollywood extended dataset (*Bojanowski et al.*, 2014) in Table 9.8 for weakly supervised action segmentation and alignment. Our approach outperforms all other methods except of CDFL for both tasks and datasets. As already

Approach	MoF - Action Segmentation	IoD - Action Alignment
ECTC *Huang et al.* (2016)	27.7	45.0
HMM/RNN *Richard et al.* (2017)	33.3	47.3
ISBA *Ding and Xu* (2018)	38.4	52.3
NNV *Richard et al.* (2018b)	43.0	-
D3TW *Chang et al.* (2019)	45.7	56.3
CDFL *Li et al.* (2019)	50.2	63.9
Ghoddoosian et al. (2022)	**51.4**	-
MuCon	48.5	66.2

Table 9.7: Comparison to the state-of-the-art on the Breakfast dataset. It is crucial to emphasize that the research presented in this chapter has been published in TPAMI 2022 (*Souri et al.*, 2022). To provide a thorough overview and foster a deeper understanding of the prevailing trends in the field, we have integrated contemporary state-of-the-art methods published at or subsequent to the publication date of this chapter.

Approach	MoF-BG - Action Segmentation	IoD - Action Alignment
ECTC *Huang et al.* (2016)	-	41.0
HMM/RNN *Richard et al.* (2017)	-	46.3
ISBA *Ding and Xu* (2018)	34.5	39.6
NNV *Richard et al.* (2018b)	-	48.7
D3TW *Chang et al.* (2019)	33.6	50.9
CDFL *Li et al.* (2019)	<u>40.6</u>	**52.9**
MuCon	**41.6**	<u>52.3</u>

Table 9.8: Comparison to the state-of-the-art on the Hollywood extended dataset. The highest score is indicated by a bold font and the second highest is underlined.

discussed in Section 9.4.4, our approach achieves either an accuracy that is comparable to CDFL or significantly better. Furthermore, our approach is 20 times faster during inference and provides thus a much better trade-off between accuracy and runtime.

9.4.5.1 Fully Supervised

As mentioned in Section 9.3.7, we can apply our approach to the fully supervised setting as well. For comparison, we use the same metrics of *Abu Farha and Gall* (2019) and *Li et al.* (2020a) namely MoF (termed accuracy by *Abu Farha and Gall* (2019)), Edit which measures predicted transcript similarity (similar to the matching score), and F1 score at different overlaps.

We first evaluate whether the mutual consistency loss $\mathcal{L}_\mu$ also improves the accuracy in a fully supervised setting. For this ablation experiment, we use split 1 of the Breakfast dataset. As it is shown in Table 9.9, the proposed mutual consistency loss improves the accuracy for all metrics. This shows that the mutual consistency loss is not only useful for weakly supervised learning, but also for fully supervised learning.

We furthermore compare our approach to other fully supervised action segmentation approaches in Table 9.10. Although our approach was designed for weakly supervised learning, it outperforms

Breakfast	F1@{10,25,50}			Edit	MoF
MuCon w/o $\mathcal{L}_\mu$	71.6	64.8	49.1	75.6	63.7
MuCon	**73.1**	**66.3**	**50.7**	**76.4**	**64.1**

Table 9.9: Effect of the MuCon loss on the accuracy in the fully supervised setting. The results are reported for split 1 of the Breakfast dataset.

Breakfast	F1@{10,25,50}			Edit	MoF
ED-TCN *Lea et al.* (2017)*	-	-	-	-	43.3
HTK *Kuehne et al.* (2017)	-	-	-	-	50.7
TCFPN *Ding and Xu* (2018)	-	-	-	-	52.0
HTK(64) *Kuehne et al.* (2016b)	-	-	-	-	56.3
GRU *Richard et al.* (2017)*	-	-	-	-	60.6
GRU+length prior *Kuehne et al.* (2020)	-	-	-	-	61.3
MS-TCN *Abu Farha and Gall* (2019)	52.6	48.1	37.9	61.7	66.3
MS-TCN++ *Li et al.* (2020a)	64.1	58.6	45.9	65.6	**67.6**
MuCon	**73.2**	**66.1**	**48.4**	**76.3**	62.8

Table 9.10: Comparison with the state-of-the-art for fully supervised action segmentation on the Breakfast dataset. (* obtained from *Ding and Xu* (2018)).

the state-of-the-art for most metrics. Only for the MoF accuracy, *Abu Farha and Gall* (2019) and *Li et al.* (2020a) perform better, but these networks use multiple stages and thus more layers.

9.4.5.2 Mixed Supervision

Since our network can be trained in a weakly as well in a fully supervised setting, we can train the network also in a mixed setting where a small percentage of the videos is fully annotated and the remaining videos are only weakly annotated by transcripts. As before, we report the average and standard deviation over 5 runs where we randomly sample the videos with frame-wise annotations for each run. The results for a varying percentage of fully annotated videos are reported in Table 9.11 and visualized in Figure 9.6. In case of 0%, the setting corresponds to weakly supervised learning and 100% corresponds to fully supervised learning. The difference in accuracy between the weakly and fully supervised cases is about 15%. While there is no significant improvement if only 1-2% of the videos are fully annotated, the accuracy increases when at least 5% of the videos are fully annotated. Having 10% of the videos fully annotated, the accuracy gap between the mixed setting and the fully supervised setting is already reduced by about 50%.

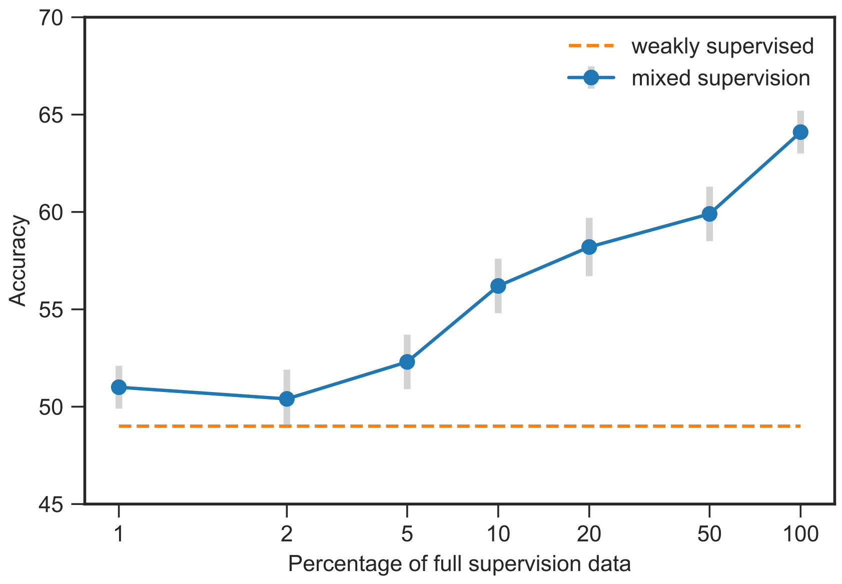
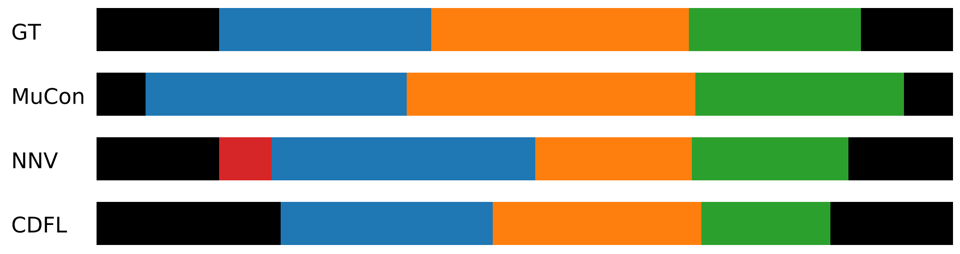


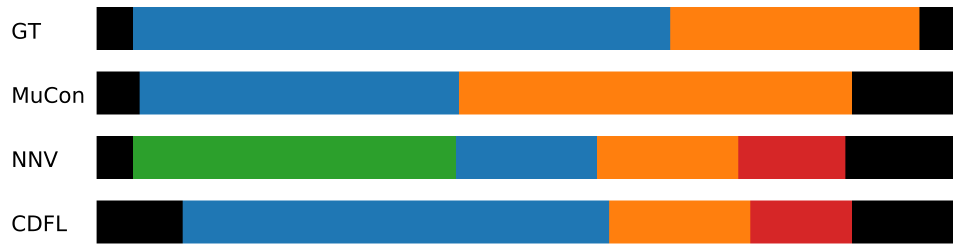
Figure 9.6: MoF accuracy (%) for training with mixed supervision. The x-axis denotes the percentage of videos that are fully annotated. The accuracy on split 1 of the Breakfast dataset is reported.

Percentage of fully supervised data $p\%$	0%	1%	2%	5%	10%	20%	50%	100%
Accuracy %	49.0	51.0	50.4	52.3	56.2	58.2	59.9	64.1

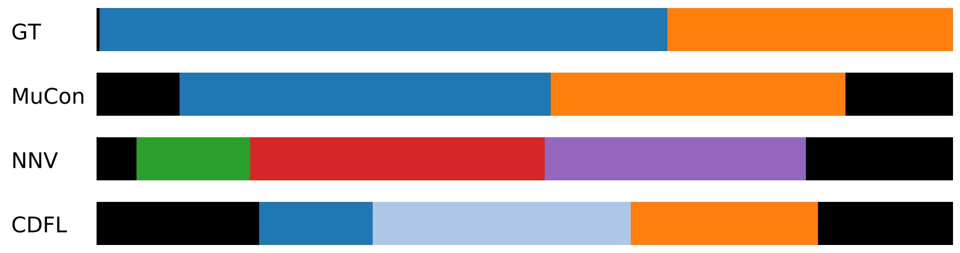
Table 9.11: Training with mixed supervision. The average MoF accuracy on split 1 of the Breakfast dataset is reported.



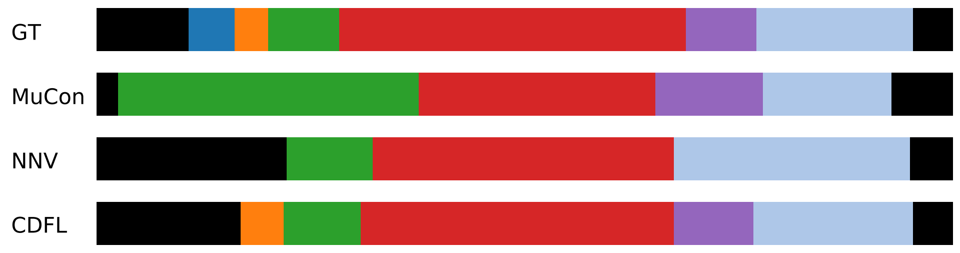
(a) c1 *background* - c2 *spoon powder* - c3 *pour milk* - c4 *stir milk* - c5 *take cup*



(b) c1 *background* - c2 *pour cereals* - c3 *pour milk* - c4 *take bowl* - c5 *stir cereals*

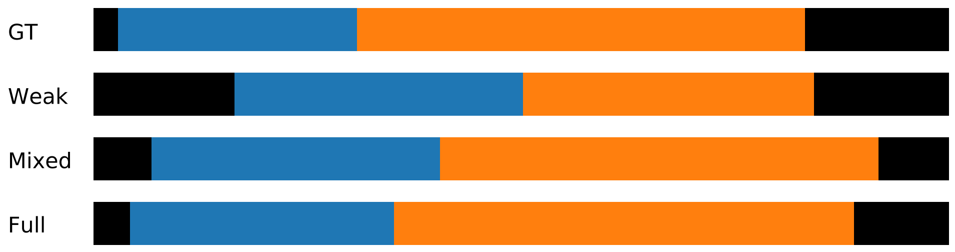


(c) c1 *background* - c2 *pour coffee* - c3 *pour milk* - c4 *take cup* - c5 *add teabag* - c6 *pour water* - c7 *spoon sugar*

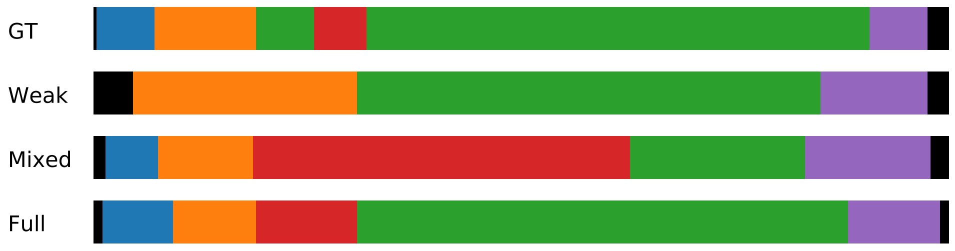


(d) c1 *background* - c2 *take plate* - c3 *take knife* - c4 *cut orange* - c5 *squeeze orange* - c6 *take glass* - c7 *pour juice*

Figure 9.7: Qualitative examples for weakly supervised action segmentation on the Breakfast dataset. Each figure visualizes a different video from the test set of split 1. We compare the results from MuCon, NNV, and CDFL with the ground truth (GT). Each row shows the result for the entire duration of a video and the colored segments show when the actions occur in the video.



(a) c1 *background* - c2 *add teabag* - c3 *pour water*



(b) c1 *background* - c2 *pour oil* - c3 *crack egg* - c4 *fry egg* - c5 *add salt and pepper* - c6 *put egg to plate*

Figure 9.8: Qualitative examples for different levels of supervision on the Breakfast dataset. Each figure visualizes a different video from the test set of split 1. *GT* visualizes the ground truth segmentation and *Weak*, *Mixed*, and *Full* visualize the output of MuCon trained with weak, mixed (10%), or full supervision, respectively.

9.4.6 Qualitative Evaluation

We provide some qualitative results for four different test videos of the Breakfast dataset in Figure 9.7. Figure 9.7a shows a video where NNV, CDFL, and MuCon perform well. Only NNV hallucinates the action *take cup*, which is not present in the video. The hallucination of actions occurs due to the alignment of the transcripts of the training set to the test video. In this example, there is a very high uncertainty among the class probabilities estimated by NNV at the beginning of the video. Since the probabilities for *take cup* are low but slightly higher than for *spoon powder* or *background*, the transcript *take cup - spoon powder - pour milk - stir milk* achieves a higher alignment score than the correct transcript *spoon powder - pour milk - stir milk*. Figures 9.7b and 9.7c show examples where also CDFL hallucinates actions that are plausible based on the transcripts but that do not occur in the video. In contrast, MuCon does not suffer from hallucinating actions since it infers the transcript directly from the test video and it does not search the training transcript that best aligns to the test video. Figure 9.7d visualizes a failure case where all approaches fail to infer the *take plate* action at the beginning of the video. In this example, CDFL provides a better estimate than MuCon.

Figure 9.8 shows qualitative examples for comparing the different types of supervision, namely weak, mixed (10%), and full supervision. Figure 9.8a shows the result for a test video where MuCon is able to infer the actions correctly using weak, mixed, or full supervision. However, the action boundaries are more accurately estimated if MuCon is trained with more supervision. Figure 9.8b shows a very difficult video where even the fully supervised approach makes a mistake.

9.5 Summary

In the preceding chapter, we introduced a weakly supervised temporal action segmentation approach that leverages merely a set of actions for training purposes. In a set of actions, only the list of actions occurring in the video is known, but not when, how often, or in which order they occur, which makes the annotation task more accessible and cost-effective. While employing a set of actions for model training is cost-effective, using it as adequate supervision for temporal action segmentation models presents a significant challenge. Considering the constraints of techniques utilizing weak action set supervision, it is crucial to explore methods that can capitalize on more robust forms of supervision. Transcripts serve as one such example of weak supervision, providing ordered lists of actions that denote the sequence in which actions transpire within a training video, albeit without specifying their temporal occurrence.

As a final contribution, in this chapter, we presented a novel approach for weakly supervised action segmentation from transcripts. The approach is based on a two-branch neural network that independently predicts two representations for action segmentation. To train the network, we introduced a new mutual consistency loss (MuCon) that enforces consistency between these two representations during training. With the use of MuCon and a transcript prediction loss, our network can be trained end-to-end without the need for additional steps. We demonstrate that the proposed network, with the mutual consistency loss, attains an accuracy that is either on par with or superior to the state-of-the-art methods. Additionally, it exhibits a marked improvement in terms of computational efficiency, offering the best trade-off between accuracy and inference time. Furthermore, our experiments demonstrate that the mutual consistency loss increases accuracy even in fully supervised learning and that the network can be applied to a mixed setting, where a few videos are fully annotated and the others are weakly annotated.

CHAPTER 10

Concolusion

In this chapter, we summarize the key contributions of this thesis. Additionally, we offer insights on promising avenues for future endeavors within this field. It is our belief that continued investigation in the realm of machine learning and computer vision will yield significant progress in the realm of video understanding. As such, we propose several potential directions for future investigation, including the incorporation of representation learning techniques and the exploration of novel spatio-temporal methods. As the field of computer vision continues to evolve, we are confident that these efforts will prove instrumental in driving further advancement.

Contents

10.1 Concolusion

In this thesis, we presented a suite of approaches that address the complex and multifaceted challenges of video understanding. Our contributions focused on addressing the difficulties inherent in modeling spatio-temporal dependencies within video data, creating a large-scale holistic video understanding dataset, making video understanding models more efficient, and training models with less supervision.

We proposed a new block, called ‘Spatio-Temporal Channel Correlation’ (STC), which models correlations between channels of a 3D CNN with respect to temporal and spatial features. By integrating this block into existing state-of-the-art architectures such as ResNext and ResNet (*Tran et al.*, 2017), we observed performance improvements of 2-3%. In addition to this contribution, we also presented a technique for transferring knowledge from pre-trained 2D CNNs to randomly initialized 3D CNNs, allowing for the efficient fine-tuning of 3D CNNs with significantly reduced amounts of training data. Our approach outperformed both generic and recent methods in the field, demonstrating the effectiveness of our proposed techniques for improving the performance of 3D CNNs in video understanding tasks.

We further focused on filling the gap of having well-established video benchmarks that integrate the simultaneous recognition of various semantic aspects within dynamic scenes. To address this gap, we introduced the "Holistic Video Understanding" Dataset (HVU), a large-scale dataset organized in a semantic taxonomy that focuses on multi-label and multi-task video understanding as a comprehensive problem. HVU contains approximately 572k videos with 9 million annotations spanning 3142 labels, encompassing categories of scenes, objects, actions, events, attributes, and concepts that capture real-world scenarios. We demonstrated the generalization capabilities of HVU on three challenging tasks: video classification, video captioning, and video clustering. For video classification, we proposed the "Holistic Appearance and Temporal Network" (HATNet), a novel spatio-temporal deep neural network architecture that fuses 2D and 3D architectures by combining intermediate representations of appearance and temporal cues, and is trained in an end-to-end manner for multi-label and multi-task learning. Our experiments validate the idea that holistic representation learning can enable many real-world applications.

Afterward, we focused on the intensive complexity of deep video models. To make 3D CNNs more efficient, we introduced a differentiable Similarity Guided Sampling (SGS) module that can be integrated into any existing 3D CNN architecture. The SGS module empowers 3D CNNs by learning the similarity between temporal features and grouping similar features together, resulting in an adaptive temporal feature resolution (ATFR) that varies for each input video clip. By integrating the SGS module into current 3D CNNs, we converted them into more efficient 3D CNNs with ATFR. Our evaluations demonstrated that the proposed module improves upon the state-of-the-art by decreasing computational cost (GFLOPs) by 50% while preserving or even improving accuracy. We evaluated the effectiveness of the SGS module by adding it to multiple state-of-the-art 3D CNNs.

Following our achievements in making 3D CNNs more efficient, we proposed a method for making vision transformers more efficient. To this end, we presented a differentiable, parameter-free Adaptive Token Sampler (ATS) module that can be integrated into any existing vision transformer architecture. The ATS module enhances vision transformers by scoring and adaptively sampling significant tokens, resulting in an adaptive number of tokens that varies for each input image or video. By incorporating the ATS module into current transformer blocks, we can convert them into more efficient vision transformers with an adaptive number of tokens. The ATS module is parameter-free, making it easy to add to off-the-shelf, pre-trained vision transformers as a plug-and-play module, reducing their computation cost without any additional training. Additionally, due to its differentiable design, a vision transformer equipped with ATS can be trained. We evaluated the efficiency of the ATS module in both image and video classification tasks by adding it to multiple state-of-the-art vision transformers. Our proposed module improved upon the state-of-the-art by reducing their computational costs by a factor of 2 while maintaining accuracy.

Finally, we tried to address the problem of video understanding beyond the low-level trimmed video understanding. To this end, we proposed two methods for temporal action segmentation with different levels of supervision. In our first method, we focused on actions set supervision. We proposed an approach that divides the video into smaller temporal regions and predicts for each region the action label and its length, as well as estimating action labels for each frame. By measuring the consistency of the frame-wise predictions with respect to the temporal regions and annotated action labels, the network learns to divide the video into class-consistent regions. We evaluated our approach on three datasets and achieve state-of-the-art results.

Despite the cost-effectiveness of utilizing a set of actions as a means of annotating videos, using it as proper supervision for temporal action segmentation models presents a daunting challenge. As a result, approaches that utilize only a set of actions as supervision are still in the early stages of development in terms of performance. As a culminating contribution, we proposed a novel end-to-end framework for weakly supervised action segmentation using transcript supervision. Our approach contains a two-branch neural network. The network's dual branches independently predict redundant yet distinct action segmentation representations, and we incorporate a mutual consistency loss term, denoted as MuCon, to enforce consistency between these redundant representations. By combining the MuCon loss with a loss function for transcript prediction, our approach attains the accuracy of state-of-the-art methods while exhibiting marked improvements in efficiency, with a factor of 14 decrease in training time and a factor of 20 increase in inference speed. Furthermore, the efficacy of the MuCon loss is also demonstrated in a fully supervised setting.

10.2 Future Work

We believe that the field of video understanding is replete with opportunities for pioneering research, particularly in the domains of spatio-temporal modeling, holistic video representation learning, efficient video processing, and long-term video understanding. In the subsequent discourse, we shall elaborate on some of these burgeoning areas of inquiry.

10.2.1 Spatio-Temporal Modeling

In this thesis, we demonstrated the effectiveness of spatio-temporal modeling by introducing our STC block, which led to improved accuracy in 3D CNNs. We posit that similar techniques can be extended to video vision transformers with the aim of enhancing their performance. Moreover, we believe that exploring alternative approaches to approximating the desired spatio-temporal model holds significant potential for advancing this research field. The majority of state-of-the-art methods focus on either convolutional neural networks or transformer neural networks with self-attention for spatio-temporal modeling. However, as demonstrated in our recent publication (*Pourheydari et al.*, 2022), we proposed a novel approach to temporal modeling through the utilization of Taylor series expansions. In this approach, we showcase that a spatio-temporal neural network, reformulated as a Taylor expansion, can better approximate the temporal dynamics in a given video compared to traditional spatio-temporal neural networks. This innovative technique resulted in enhanced outcomes for future video frame prediction, and more importantly, exhibited a higher degree of generalization. We believe that the generalization capabilities of this technique have the potential to be employed in other video understanding tasks, leading to valuable advancements in this field.

10.2.2 Holistic Video Representation

As we have demonstrated in this thesis, video understanding, which encompasses the recognition of multiple semantic aspects such as scene or environment, objects, actions, events, attributes, and concepts, remains a challenging problem in computer vision despite the significant progress made in action recognition. The lack of established video benchmarks that integrate the joint recognition of multiple semantic aspects in dynamic scenes has been identified as a major contributing factor to this challenge. To address this limitation, we presented our Holistic Video Understanding (HVU) dataset, which has been designed to alleviate this limitation by providing a comprehensive and diverse dataset that supports the joint recognition of multiple semantic aspects. Our experimental results have demonstrated that training a video understanding model on such a dataset with richer annotations improves the quality of the model's representation, which can also be beneficial for other downstream tasks, such as multi-label video classification. Recently, there has been a new line of research that focuses on self-supervised video representation methods, which have been shown to improve representation quality by evaluating the methods on downstream tasks such as action recognition. However, these methods have failed to provide empirical evidence to support their claims and motivations. In light of this, we believe that our HVU dataset can play a vital role in these self-supervised video representation methods. By using the HVU dataset, these methods can evaluate their representation learning techniques not only on action recognition but also on other categories available in our dataset. Thus, providing a comprehensive and diverse benchmark that can support the evaluation of these methods. Furthermore, we believe that the research community can benefit from our dataset not only for studying different aspects of richer annotation for training video understanding models but also as an important benchmark in other areas, such as self-supervised video representation learning.

10.2.3 Efficient Methods

As the sophistication of video understanding models increases, so too do their computational demands. In this thesis, we presented methods that dynamically adapt their computational resources to the content of the input video data. However, these methods currently face limitations in terms of hardware support. Thus, it is imperative that further research is conducted in this direction to fully realize the potential for computation cost reduction through adaptive methods, and to facilitate the development of appropriate future hardware support. Another promising area for the application of computational efficiency enhancement techniques is in other video processing tasks, such as video semantic segmentation, where significant improvements may be achieved. Additionally, the absence of a standardized efficiency metric for neural networks hinders the proper comparison of the processing costs of different models, highlighting the need for the development of a comprehensive metric to address this issue. In the following, we provide a thorough analysis of potential future directions to address these key factors.

10.2.3.1 Adaptive Models

One of the defining characteristics of human cognition is the ability to adapt biological resources to the complexity of a task at hand. In this thesis, we demonstrated the feasibility of designing models that can adapt their computation resources to the complexity of their input data. We established that there is a significant scope for enhancing the efficiency of video understanding models by judiciously allocating computation resources to avoid redundant processing. However, as evidenced by our empirical results, the actual rate of speedup on current hardware is less than the theoretical improvement due to factors arising from hardware design. Current hardware is optimized for batch processing of static neural networks, which do not dynamically alter intermediate feature maps. Therefore, we encourage the research community to continue investigating this line of research to demonstrate the potential of adaptive models to the industry. This would lead to the development of dynamic neural network-friendly hardware in the near future, resulting in more efficient models that consume less energy and are capable of more processing given the same computation resources, which could be widely available in industrial products.

10.2.3.2 Efficient Video Processing

Video processing is a multifaceted field of study that encompasses various tasks, including but not limited to, object detection and tracking, video semantic segmentation, and video enhancement. However, current state-of-the-art approaches tend to treat these problems as if they were image processing problems, instead of fully leveraging the temporal nature of video data. Specifically, they process video as a sequence of individual frames, rather than as a holistic representation. This approach, however, is inherently suboptimal, as it leads to the processing of a large amount of redundant data. As previously discussed, video data is replete with redundancies, and therefore, processing individual frames individually is not the most efficient approach to video processing. As demonstrated in our thesis, we were able to significantly improve the efficiency of a video understanding model by avoiding the processing of redundant data. Consequently, we believe that similar approaches could be applied to other video processing tasks, with the aim of improving their efficiency.

10.2.3.3 Efficiency Metrics

The currently prevalent metric utilized to gauge the efficiency of neural networks is GFLOP, which provides a relatively accurate approximation of the floating point operations performed within a neural network. However, as this metric was designed with static neural networks in mind, it fails to take into account several crucial factors that ultimately impact the actual processing cost of a neural network. One such important factor is the number of memory input/output (IO) operations, as memory access is typically a computationally expensive task. The number of such operations plays a crucial role in determining the cost of a neural network. Additionally, the metric also neglects the number of iterative procedures performed within a neural network. For example, the implementation of an expensive iterative clustering method inside a neural network to aggregate feature maps and subsequently reduce GFLOPs would yield a lower GFLOPs value, but it does not truly reflect the cost of the neural network. Therefore, we believe that the research community should strive to develop a more comprehensive metric that captures these missing factors, which would provide a more accurate representation of the actual cost of a neural network.

10.2.4 Long term video understanding

Experience and reasoning are known to encompass multiple temporal scales, ranging from milliseconds to days. Despite this, the majority of computer vision research remains focused on individual images or brief videos lasting only a few seconds. In this thesis, we demonstrate methods that are capable of processing long-term videos by utilizing pretrained models to represent the videos as extracted features. These models were all trained on supervised action recognition tasks, which necessitate large amounts of annotated data. Therefore, we believe that these models could be replaced with models trained on self-supervised tasks. Another significant challenge in long-term video understanding is the limitations imposed by memory. In our opinion, this constitutes a key obstacle in the development of models that can directly process the given video without the need for any pre-processing feature extraction steps. Additionally, the lack of large-scale, long-term video understanding datasets constitutes a significant bottleneck in the development of high-quality long-term video understanding methods. In the following, we provide a thorough analysis of potential future directions to address these key factors.

10.2.4.1 Memory Limitations

A prevalent challenge in the domain of long-term video understanding is the scalability of methods required to process longer videos. As video data is a sequence of still images, it necessitates a substantial amount of storage space and processing memory. This high volume of data results in various issues such as data transfer bottlenecks, speed limitations, and memory constraints. To conserve storage space, video data is commonly stored in a compressed format. However, this compression has the adverse effect of increasing the computational power required for decompression, thus slowing down the video processing pipeline. Current training pipelines propose storing videos in a decompressed format to alleviate this issue, yet this approach has its own drawbacks such as increased storage usage and saturation of the processors' data transfer lines. We believe that there are multiple approaches to mitigate these challenges. One potential solution could involve unifying the video processing pipeline by compressing the video data using neural compression techniques and processing the video using the same compressed data, thus eliminating the need for decompression. An alternate approach could involve designing neural networks that can handle compressed video data from current standard codecs. Such techniques, unlike the previous one mentioned, would be beneficial in terms of consistency over current standardized codecs. However, these techniques may introduce new limitations, such as limitations in data augmentation, which could be resolved in future research. In conclusion, we believe that addressing the memory constraints currently facing the field of video understanding can pave the way for the development of approaches that are capable of processing long-term videos in an end-to-end fashion. This, in turn, could open up new horizons for research in video understanding, allowing for the examination of more complex and nuanced phenomena, and ultimately leading to more powerful and sophisticated video understanding systems.

10.2.4.2 Feature Learning

Current methodologies for temporal action segmentation and untrimmed video understanding generally leverage a pretrained model as a feature extractor for the input video. These pretrained models are typically trained on supervised tasks, which necessitate a vast amount of annotated data. Recent advancements in self-supervised approaches for pretraining video understanding models have emerged, which can be applied to downstream tasks, such as trimmed video action recognition. In one of our recent publications (*Behrmann et al.*, 2021), we proposed a self-supervised method for pretraining video understanding models. Our work demonstrates that a backbone pretrained using our self-supervised approach is capable of extracting superior features compared to a model pretrained on supervised action recognition tasks. Furthermore, we show that employing these methods for a downstream task, such as fully supervised temporal action segmentation on long-term videos, not only improves performance compared to fully supervised features but also reduces the need for supervision during the training of the feature extraction model. Given these promising results, we believe that utilizing similar techniques for pretraining feature extraction models could also be beneficial for weakly supervised temporal action segmentation models. In light of these considerations, we encourage the research community to undertake a more comprehensive examination of the impact of various feature learning techniques on long-term video understanding methodologies. Such investigations have the potential to significantly reduce the reliance on supervision, specifically in the context of weakly supervised temporal action segmentation methodologies. Furthermore, by leveraging large amounts of unlabeled video data, these techniques may enable the learning of more universal and generalizable representations, thereby enhancing the performance of long-term video understanding methods.

10.2.4.3 Larger Datasets

The dearth of large-scale, long-term video understanding datasets constitutes a formidable obstacle in the advancement of state-of-the-art long-term video understanding methodologies. The absence of comprehensive and diverse datasets hinders the ability to train and evaluate models on a broad range of scenarios, limiting the generalizability and robustness of the developed models. For instance, the Breakfast dataset (*Kuehne et al.*, 2014), which is a widely-utilized and prominent dataset in temporal action segmentation, is recorded in a confined setting and comprises a relatively small number of videos, at around 1700. The limited size of such datasets poses a significant challenge for temporal action segmentation models in their ability to learn to accurately predict transcripts without overfitting. Furthermore, models trained on the Breakfast dataset demonstrate restricted applicability, as they fail to generalize to other environments, emphasizing the imperative for more diverse and comprehensive datasets to be developed. This necessitates the curation of novel and more varied long-term video understanding datasets, which can foster the advancement of this field of study. The generation of such datasets can provide the necessary resources for training and evaluating models on a broad range of scenarios, enabling the generalizability and robustness of the developed models, and ultimately leading to a more comprehensive understanding of long-term video understanding.

Bibliography

Google vision ai api. `cloud.google.com/vision`. (Cited on page 54.)

Sensifai video tagging api. `www.sensifai.com`. (Cited on page 54.)

Abu-El-Haija, Sami; Kothari, Nisarg; Lee, Joonseok; Natsev, Paul; Toderici, George; Varadarajan, Balakrishnan, and Vijayanarasimhan, Sudheendra. Youtube-8m: A large-scale video classification benchmark. *arXiv:1609.08675*, 2016. (Cited on pages 43 and 53.)

Abu Farha, Yazan and Gall, Juergen. Ms-tcn: Multi-stage temporal convolutional network for action segmentation. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019. (Cited on pages 4, 19, 24, 26, 32, 116, 134, 139, 141, 142, 146 and 147.)

Ahn, Hyemin and Lee, Dongheui. Refining action segmentation with hierarchical video representations. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, October 2021. (Cited on page 32.)

Arnab, Anurag; Dehghani, Mostafa; Heigold, Georg; Sun, Chen; Lučić, Mario, and Schmid, Cordelia. Vivit: A video vision transformer. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 6836–6846, October 2021. (Cited on pages 62, 112 and 113.)

Bahdanau, Dzmitry; Cho, Kyunghyun, and Bengio, Yoshua. Neural machine translation by jointly learning to align and translate. In *International Conference on Learning Representations (ICLR)*, 2015. (Cited on page 134.)

Behrmann, Nadine; Fayyaz, Mohsen; Gall, Juergen, and Noroozi, Mehdi. Long short view feature decomposition via contrastive video representation learning. In *IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 9244–9253, October 2021. (Cited on page 158.)

Bengio, Yoshua; Simard, Patrice, and Frasconi, Paolo. Learning long-term dependencies with gradient descent is difficult. *IEEE transactions on neural networks*, pages 157–166, 1994. (Cited on page 16.)

Bertasius, Gedas; Wang, Heng, and Torresani, Lorenzo. Is space-time attention all you need for video understanding. In *International Conference on Machine Learning (ICML)*, 2021. (Cited on pages 3, 22, 30, 45, 62, 90, 92, 107, 109, 112 and 113.)

Bojanowski, Piotr; Lajugie, Rémi; Bach, Francis; Laptev, Ivan; Ponce, Jean; Schmid, Cordelia, and Sivic, Josef. Weakly supervised action labeling in videos under ordering constraints. In *European Conference on Computer Vision (ECCV)*, 2014. (Cited on pages 28, 33, 124, 141 and 145.)

Bulat, Adrian; Perez Rua, Juan Manuel; Sudhakaran, Swathikiran; Martinez, Brais, and Tzimiropoulos, Georgios. Space-time mixing attention for video transformer. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2021. (Cited on pages xv, xvi, 3, 22, 30, 45, 62, 90, 92, 105, 107, 109, 112, 113 and 126.)

Caba Heilbron, Fabian; Escorcia, Victor; Ghanem, Bernard, and Carlos Niebles, Juan. Activitynet: A large-scale video benchmark for human activity understanding. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015. (Cited on pages 53 and 63.)

Cai, Zhuowei; Wang, Limin; Peng, Xiaojiang, and Qiao, Yu. Multi-view super vector for action recognition. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2014. (Cited on page 46.)

Carion, Nicolas; Massa, Francisco; Synnaeve, Gabriel; Usunier, Nicolas; Kirillov, Alexander, and Zagoruyko, Sergey. End-to-end object detection with transformers. In *European Conference on Computer Vision (ECCV)*, 2020. (Cited on page 30.)

Carreira, Joao and Zisserman, Andrew. Quo vadis, action recognition? a new model and the kinetics dataset. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017. (Cited on pages 2, 19, 24, 28, 30, 43, 45, 46, 62, 70, 73, 78, 83, 84, 124, 126 and 141.)

Carreira, João; Noland, Eric; Banki-Horvath, Andras; Hillier, Chloe, and Zisserman, Andrew. A short note about kinetics-600. In *arXiv preprint arXiv:1808.01340v1*, 2018. (Cited on pages 53, 73, 79, 95, 107 and 108.)

Chang, Chien-Yi; Huang, De-An; Sui, Yanan; Fei-Fei, Li, and Niebles, Juan Carlos. D^{3}TW: Discriminative differentiable dynamic time warping for weakly supervised action alignment and segmentation. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019. (Cited on pages 4, 116, 130, 141 and 146.)

Chen, Chun-Fu; Fan, Quanfu, and Panda, Rameswar. Crossvit: Cross-attention multi-scale vision transformer for image classification. In *IEEE/CVF International Conference on Computer Vision (ICCV)*, 2021. (Cited on page 111.)

Chen, Shaoxiang and Jiang, Yu-Gang. Motion guided spatial attention for video captioning. In *AAAI*, 2019. (Cited on page 63.)

Chen, Yunpeng; Kalantidis, Yannis; Li, Jianshu; Yan, Shuicheng, and Feng, Jiashi. Aˆ 2-nets: Double attention networks. In *Advances in neural information processing systems (NeuRIPS)*, pages 352–361, 2018. (Cited on page 81.)

Cheng, Bowen; Schwing, Alexander G., and Kirillov, Alexander. Per-pixel classification is not all you need for semantic segmentation. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2021. (Cited on page 30.)

Child, Rewon; Gray, Scott; Radford, Alec, and Sutskever, Ilya. Generating long sequences with sparse transformers. In *arXiv preprint arXiv:1904.10509*, 2019. (Cited on page 31.)

Cho, Kyunghyun; van Merrienboer, B; Gulcehre, Caglar; Bougares, F; Schwenk, H, and Bengio, Yoshua. Learning phrase representations using rnn encoder-decoder for statistical machine translation. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2014a. (Cited on page 16.)

Cho, Kyunghyun; van Merrienboer, Bart; Bahdanau, Dzmitry, and Bengio, Yoshua. On the properties of neural machine translation: Encoder–decoder approaches. In *Workshop on Syntax, Semantics and Structure in Statistical Translation*, 2014b. (Cited on page 134.)

Chu, Xiangxiang; Tian, Zhi; Zhang, Bo; Wang, Xinlong; Wei, Xiaolin; Xia, Huaxia, and Shen, Chunhua. Conditional positional encodings for vision transformers. In *arXiv preprint arXiv:2102.10882*, 2021. (Cited on page 111.)

Dalal, Navneet; Triggs, Bill, and Schmid, Cordelia. Human detection using oriented histograms of flow and appearance. In *European Conference on Computer Vision (ECCV)*, 2006. (Cited on page 29.)

Damen, Dima; Doughty, Hazel; Maria Farinella, Giovanni; Fidler, Sanja; Furnari, Antonino; Kazakos, Evangelos; Moltisanti, Davide; Munro, Jonathan; Perrett, Toby; Price, Will, and others, . Scaling egocentric vision: The epic-kitchens dataset. In *European Conference on Computer Vision and Pattern Recognition (ECCV)*, 2018. (Cited on page 53.)

Deng, Jia; Dong, Wei; Socher, Richard; Li, Li-Jia; Li, Kai, and Fei-Fei, Li. Imagenet: A large-scale hierarchical image database. In *IEEE conference on computer vision and pattern recognition (CVPR)*, pages 248–255, 2009. (Cited on pages 2, 24, 40, 42, 95 and 107.)

Diba, Ali; Sharma, Vivek, and Van Gool, Luc. Deep temporal linear encoding networks. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017. (Cited on pages 2, 29, 36 and 68.)

Diba, Ali; Fayyaz, Mohsen; Sharma, Vivek; Arzani, M Mahdi; Yousefzadeh, Rahman; Gall, Juergen, and Van Gool, Luc. Spatio-temporal channel correlation networks for action classification. In *European Conference on Computer Vision (ECCV)*, 2018. (Cited on pages xiii, 9, 45, 68 and 90.)

Diba, Ali; Sharma, Vivek; Gool, Luc Van, and Stiefelhagen, Rainer. Dynamonet: Dynamic action and motion network. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 6192–6201, 2019. (Cited on pages 30, 62, 68, 84 and 90.)

Diba, Ali; Fayyaz, Mohsen; Sharma, Vivek; Paluri, Manohar; Gall, Jürgen; Stiefelhagen, Rainer, and Van Gool, Luc. Large scale holistic video understanding. In *European Conference on Computer Vision (ECCV)*, 2020. (Cited on pages xiv, 9, 62 and 84.)

Ding, Li and Xu, Chenliang. Weakly-supervised action segmentation with iterative soft boundary assignment. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018. (Cited on pages xvii, 4, 5, 26, 33, 116, 130, 131, 133, 141, 142, 144, 146 and 147.)

Dosovitskiy, Alexey; Beyer, Lucas; Kolesnikov, Alexander; Weissenborn, Dirk; Zhai, Xiaohua; Unterthiner, Thomas; Dehghani, Mostafa; Minderer, Matthias; Heigold, Georg; Gelly, Sylvain; Uszkoreit, Jakob, and Houlsby, Neil. An image is worth 16x16 words: Transformers for image recognition at scale. In *International Conference on Learning Representations (ICLR)*, 2021. (Cited on pages 3, 21, 30, 90 and 111.)

Fan, Haoqi; Xiong, Bo; Mangalam, Karttikeya; Li, Yanghao; Yan, Zhicheng; Malik, Jitendra, and Feichtenhofer, Christoph. Multiscale vision transformers. In *IEEE/CVF International Conference on Computer Vision (ICCV)*, 2021a. (Cited on page 30.)

Fan, Haoqi; Xiong, Bo; Mangalam, Karttikeya; Li, Yanghao; Yan, Zhicheng; Malik, Jitendra, and Feichtenhofer, Christoph. Multiscale vision transformers. In *IEEE/CVF International Conference on Computer Vision (ICCV)*, 2021b. (Cited on pages 45, 62, 112 and 113.)

Fan, Quanfu; Chen, Chun-Fu (Ricarhd); Kuehne, Hilde; Pistoia, Marco, and Cox, David. More Is Less: Learning Efficient Video Representations by Temporal Aggregation Modules. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2019. (Cited on page 112.)

Fayyaz, Mohsen and Gall, Jurgen. Sct: Set constrained temporal transformer for set supervised action segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020. (Cited on pages xvi, 9, 24, 127 and 134.)

Fayyaz, Mohsen; Bahrami, Emad; Diba, Ali; Noroozi, Mehdi; Adeli, Ehsan; Van Gool, Luc, and Gall, Jurgen. 3d cnns with adaptive temporal feature resolutions. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2021. (Cited on pages xiv, xv, 9, 83 and 84.)

Fayyaz, Mohsen; Koohpayegani, Soroush Abbasi; Jafari, Farnoush Rezaei; Sengupta, Sunando; Joze, Hamid Reza Vaezi; Sommerlade, Eric; Pirsiavash, Hamed, and Gall, Jürgen. Eco: Efficient convolutional network for online video understanding. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2022. (Cited on pages xvi, 9, 112, 113 and 126.)

Feichtenhofer, Christoph. X3d: Expanding architectures for efficient video recognition. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020. (Cited on pages xiv, 30, 31, 45, 62, 69, 70, 74, 77, 78, 79, 83, 84, 90, 112 and 113.)

Feichtenhofer, Christoph; Pinz, Axel, and Zisserman, Andrew. Convolutional two-stream network fusion for video action recognition. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. (Cited on pages 29 and 46.)

Feichtenhofer, Christoph; Fan, Haoqi; Malik, Jitendra, and He, Kaiming. Slowfast networks for video recognition. *IEEE/CVF International Conference on Computer Vision (ICCV)*, 2019. (Cited on pages 30, 31, 45, 58, 62, 68, 74, 78, 79, 83, 84, 90, 112 and 113.)

Fernando, Basura; Gavves, Efstratios; Oramas, Jose M; Ghodrati, Amir, and Tuytelaars, Tinne. Modeling video evolution for action recognition. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015. (Cited on page 29.)

Gaidon, Adrien; Harchaoui, Zaid, and Schmid, Cordelia. Temporal localization of actions with actoms. *PAMI*, 35(11):2782–2795, 2013. (Cited on page 32.)

Gao, Shang-Hua; Han, Qi; Li, Zhong-Yu; Peng, Pai; Wang, Liang, and Cheng, Ming-Ming. Global2local: Efficient structure search for video action segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 16805–16814, 2021. (Cited on page 32.)

Ghoddoosian, Reza; Sayed, Saif, and Athitsos, Vassilis. Hierarchical modeling for task recognition and action segmentation in weakly-labeled instructional videos. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, January 2022. (Cited on page 146.)

Goyal, Priya; Dollár, Piotr; Girshick, Ross B.; Noordhuis, Pieter; Wesolowski, Lukasz; Kyrola, Aapo; Tulloch, Andrew; Jia, Yangqing, and He, Kaiming. Accurate, large minibatch sgd: Training imagenet in 1 hour. *ArXiv*, 2017a. (Cited on page 74.)

Goyal, Raghav; Kahou, Samira Ebrahimi; Michalski, Vincent; Materzynska, Joanna; Westphal, Susanne; Kim, Heuna; Haenel, Valentin; Fruend, Ingo; Yianilos, Peter; Mueller-Freitag, Moritz, and others, . The" something something" video database for learning and evaluating visual common sense. In *IEEE/CVF International Conference on Computer Vision (ICCV)*, 2017b. (Cited on pages 53, 73 and 79.)

Goyal, Saurabh; Choudhury, Anamitra R.; Raje, Saurabh M.; Chakaravarthy, Venkatesan T.; Sabharwal, Yogish, and Verma, Ashish. Power-bert: Accelerating bert inference via progressive word-vector elimination. In *International Conference on Machine Learning (ICML)*, 2020. (Cited on page 31.)

Gu, Chunhui; Sun, Chen; Ross, David A; Vondrick, Carl; Pantofaru, Caroline; Li, Yeqing; Vijayanarasimhan, Sudheendra; Toderici, George; Ricco, Susanna; Sukthankar, Rahul; Schmid, Cordelia, and Malik, Jitendra. Ava: A video dataset of spatio-temporally localized atomic visual actions. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018. (Cited on pages 51 and 53.)

Guo, Qipeng; Qiu, Xipeng; Liu, Pengfei; Shao, Yunfan; Xue, Xiangyang, and Zhang, Zheng. Star-transformer. In *arXiv preprint arXiv:1902.09113*, 2019. (Cited on page 31.)

Han, Kai; Xiao, An; Wu, Enhua; Guo, Jianyuan; Xu, Chunjing, and Wang, Yunhe. Transformer in transformer. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2021. (Cited on page 111.)

Hara, Kensho; Kataoka, Hirokatsu, and Satoh, Yutaka. Can spatiotemporal 3D CNNs retrace the history of 2D CNNs and imagenet? In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018. (Cited on pages 5, 36, 37, 43, 45 and 62.)

He, Kaiming; Zhang, Xiangyu; Ren, Shaoqing, and Sun, Jian. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *2017 IEEE International Conference on Computer Vision (ICCV)*, 2015. (Cited on pages 39, 42, 43 and 44.)

He, Kaiming; Zhang, Xiangyu; Ren, Shaoqing, and Sun, Jian. Deep residual learning for image recognition. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016a. (Cited on pages 40 and 90.)

He, Kaiming; Zhang, Xiangyu; Ren, Shaoqing, and Sun, Jian. Deep residual learning for image recognition. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016b. (Cited on page 18.)

Hochreiter, Sepp and Schmidhuber, Jürgen. Long short-term memory. *Neural computation*, pages 1735–1780, 1997. (Cited on pages 16 and 17.)

Hu, Jie; Shen, Li, and Sun, Gang. Squeeze-and-excitation networks. *arXiv preprint arXiv:1709.01507*, 2017. (Cited on page 37.)

Huang, De-An; Fei-Fei, Li, and Niebles, Juan Carlos. Connectionist temporal modeling for weakly supervised action labeling. In *European Conference on Computer Vision (ECCV)*, 2016. (Cited on pages 4, 33, 116, 130 and 146.)

Huang, Yifei; Sugano, Yusuke, and Sato, Yoichi. Improving action segmentation via graph-based temporal reasoning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020. (Cited on page 32.)

Ishikawa, Yuchi; Kasai, Seito; Aoki, Yoshimitsu, and Kataoka, Hirokatsu. Alleviating over-segmentation errors by detecting action boundaries. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, pages 2322–2331, 2021. (Cited on page 32.)

Jaderberg, Max; Simonyan, Karen; Zisserman, Andrew, and others, . Spatial transformer networks. In *NeurIPS*, 2015. (Cited on pages 120 and 138.)

Jaszczur, Sebastian; Chowdhery, Aakanksha; Mohiuddin, Afroz; Kaiser, Lukasz; Gajewski, Wojciech; Michalewski, Henryk, and Kanerva, Jonni. Sparse is enough in scaling transformers. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2021. (Cited on page 31.)

Jiang, Boyuan; Wang, MengMeng; Gan, Weihao; Wu, Wei, and Yan, Junjie. Stm: Spatiotemporal and motion encoding for action recognition. In *IEEE/CVF International Conference on Computer Vision (ICCV)*, 2019. (Cited on page 112.)

Jiang, Zihang; Hou, Qibin; Yuan, Li; Zhou, Daquan; Shi, Yujun; Jin, Xiaojie; Wang, Anran, and Feng, Jiashi. All tokens matter: Token labeling for training better vision transformers. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2021. (Cited on pages 3, 30, 90 and 111.)

Jiao, Xiaoqi; Yin, Yichun; Shang, Lifeng; Jiang, Xin; Chen, Xiao; Li, Linlin; Wang, Fang, and Liu, Qun. Tinybert: Distilling bert for natural language understanding. In *arXiv preprint arXiv:1909.10351*, 2020. (Cited on page 31.)

Karaman, Svebor; Seidenari, Lorenzo, and Del Bimbo, Alberto. Fast saliency based pooling of fisher encoded dense trajectories. In *European Conference on Computer Vision (ECCV), Workshops*, 2014. (Cited on page 32.)

Karpathy, Andrej; Toderici, George; Shetty, Sanketh; Leung, Thomas; Sukthankar, Rahul, and Fei-Fei, Li. Large-scale video classification with convolutional neural networks. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2014a. (Cited on pages 1 and 24.)

Karpathy, Andrej; Toderici, George; Shetty, Sanketh; Leung, Thomas; Sukthankar, Rahul, and Fei-Fei, Li. Large-scale video classification with convolutional neural networks. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2014b. (Cited on page 29.)

Karpathy, Andrej; Toderici, George; Shetty, Sanketh; Leung, Thomas; Sukthankar, Rahul, and Fei-Fei, Li. Large-scale video classification with convolutional neural networks. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2014c. (Cited on pages 30 and 36.)

Kay, Will; Carreira, Joao; Simonyan, Karen; Zhang, Brian; Hillier, Chloe; Vijayanarasimhan, Sudheendra; Viola, Fabio; Green, Tim; Back, Trevor; Natsev, Paul, and others, . The kinetics human action video dataset. *arXiv preprint arXiv:1705.06950*, 2017. (Cited on pages 24, 28, 30, 36, 41, 42, 51, 73, 78, 95, 107, 108, 124 and 141.)

Klaser, Alexander; Marszałek, Marcin, and Schmid, Cordelia. A spatio-temporal descriptor based on 3d-gradients. In *BMVC*, 2008. (Cited on page 29.)

Korbar, Bruno; Tran, Du, and Torresani, Lorenzo. Scsampler: Sampling salient clips from video for efficient action recognition. In *IEEE/CVF International Conference on Computer Vision (ICCV)*, 2019. (Cited on pages 2, 31, 69, 70, 79 and 83.)

Krizhevsky, Alex; Sutskever, Ilya, and Hinton, Geoffrey E. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems (NeuRIPS)*, 2012. (Cited on pages 18 and 90.)

Kuehne, H.; Richard, A., and Gall, J. A Hybrid RNN-HMM Approach for Weakly Supervised Temporal Action Segmentation. *PAMI*, 42(4):765–779, 2020. (Cited on pages 33 and 147.)

Kuehne, Hilde; Arslan, Ali, and Serre, Thomas. The language of actions: Recovering the syntax and semantics of goal-directed human activities. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2014. (Cited on pages xi, xvii, 28, 124, 131, 141, 142, 144, 145 and 159.)

Kuehne, Hilde; Gall, Juergen, and Serre, Thomas. An end-to-end generative framework for video segmentation and recognition. In *2016 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 1–8, 2016a. (Cited on pages 23, 24 and 26.)

Kuehne, Hilde; Gall, Juergen, and Serre, Thomas. An end-to-end generative framework for video segmentation and recognition. In *WACV*, 2016b. (Cited on pages 4, 32, 116, 141 and 147.)

Kuehne, Hilde; Richard, Alexander, and Gall, Juergen. Weakly supervised learning of actions from transcripts. *CVIU*, 2017. (Cited on pages 33, 116 and 147.)

Kuehne, Hildegard; Jhuang, Hueihan; Garrote, Estíbaliz; Poggio, Tomaso, and Serre, Thomas. Hmdb: a large video database for human motion recognition. In *IEEE/CVF International Conference on Computer Vision (ICCV)*, 2011. (Cited on pages 27, 36, 41, 42, 51, 53, 73 and 80.)

Lambert, Ben. edit-distance. `https://github.com/belambert/edit-distance`, 2019. (Cited on pages 26 and 141.)

Laptev, Ivan; Marszalek, Marcin; Schmid, Cordelia, and Rozenfeld, Benjamin. Learning realistic human actions from movies. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2008. (Cited on page 29.)

Lea, Colin; Reiter, Austin; Vidal, René, and Hager, Gregory D. Segmental spatiotemporal cnns for fine-grained action segmentation. In *European Conference on Computer Vision (ECCV)*, 2016. (Cited on page 32.)

Lea, Colin; Flynn, Michael D; Vidal, Rene; Reiter, Austin, and Hager, Gregory D. Temporal convolutional networks for action segmentation and detection. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017. (Cited on pages 4, 32, 116, 134 and 147.)

Lecun, Y.; Bottou, L.; Bengio, Y., and Haffner, P. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, pages 2278–2324, 1998. (Cited on page 18.)

LeCun, Yann. A path towards autonomous machine intelligence version 0.9.2, 2022-06-27. In *OpenReview*, 2022. (Cited on page 1.)

Lee, Myunggi; Lee, Seungeui; Son, Sungjoon; Park, Gyutae, and Kwak, Nojun. Motion feature network: Fixed motion filter for action recognition. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 387–403, 2018. (Cited on page 30.)

Li, Jun and Todorovic, Sinisa. Set-constrained viterbi for set-supervised action segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020. (Cited on pages 32 and 127.)

Li, Jun and Todorovic, Sinisa. Anchor-constrained viterbi for set-supervised action segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 9806–9815, 2021. (Cited on page 32.)

Li, Jun; Lei, Peng, and Todorovic, Sinisa. Weakly supervised energy-based learning for action segmentation. In *IEEE/CVF International Conference on Computer Vision (ICCV)*, 2019. (Cited on pages 4, 5, 26, 33, 116, 130, 131, 132, 140, 141, 142, 144, 145 and 146.)

Li, Shi-Jie; AbuFarha, Yazan; Liu, Yun; Cheng, Ming-Ming, and Gall, Juergen. MS-TCN++: Multi-Stage Temporal Convolutional Network for Action Segmentation. *PAMI*, 2020a. (Cited on pages 24, 32, 134, 141, 142, 146 and 147.)

Li, Yan; Ji, Bin; Shi, Xintian; Zhang, Jianguo; Kang, Bin, and Wang, Limin. Tea: Temporal excitation and aggregation for action recognition. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020b. (Cited on page 112.)

Li, Yanghao; Wu, Chao-Yuan; Fan, Haoqi; Mangalam, Karttikeya; Xiong, Bo; Malik, Jitendra, and Feichtenhofer, Christoph. Mvitv2: Improved multiscale vision transformers for classification and detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4804–4814, June 2022. (Cited on page 30.)

Liang, Youwei; Ge, Chongjian; Tong, Zhan; Song, Yibing; Wang, Jue, and Xie, Pengtao. Not all patches are what you need: Expediting vision transformers via token reorganizations. In *International Conference on Learning Representations (ICLR)*, 2022. (Cited on pages xv, 3, 32, 91, 92, 104, 107 and 111.)

Lin, Ji; Gan, Chuang, and Han, Song. Tsm: Temporal shift module for efficient video understanding. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 7083–7093, 2019. (Cited on pages 29, 30, 83 and 112.)

Liu, Sheng; Ren, Zhou, and Yuan, Junsong. Sibnet: Sibling convolutional encoder for video captioning. In *ACMM*, 2018. (Cited on page 63.)

Liu, Ze; Lin, Yutong; Cao, Yue; Hu, Han; Wei, Yixuan; Zhang, Zheng; Lin, Stephen, and Guo, Baining. Swin transformer: Hierarchical vision transformer using shifted windows. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2021. (Cited on pages 3, 30, 90 and 111.)

Liu, Ze; Ning, Jia; Cao, Yue; Wei, Yixuan; Zhang, Zheng; Lin, Stephen, and Hu, Han. Video swin transformer. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022. (Cited on pages 83, 84, 112 and 113.)

Loshchilov, Ilya and Hutter, Frank. Sgdr: Stochastic gradient descent with warm restarts. In *International Conference on Learning Representations (ICLR)*, 2017. (Cited on page 74.)

Lu, Zijia and Elhamifar, Ehsan. Set-supervised action learning in procedural task videos via pairwise order consistency. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022. (Cited on page 127.)

Marin, Dmitrii; Chang, Jen-Hao Rick; Ranjan, Anurag; Prabhu, Anish K.; Rastegari, Mohammad, and Tuzel, Oncel. Token pooling in vision transformers. *arXiv preprint arXiv:2110.03860*, 2021. (Cited on pages 32 and 108.)

Miller, George A. Wordnet: a lexical database for english. *Communications of the ACM*, 1995. (Cited on page 55.)

Monfort, Mathew; Andonian, Alex; Zhou, Bolei; Ramakrishnan, Kandan; Bargal, Sarah Adel; Yan, Tom; Brown, Lisa; Fan, Quanfu; Gutfreund, Dan; Vondrick, Carl, and others, . Moments in time dataset: one million videos for event understanding. *IEEE transactions on pattern analysis and machine intelligence (TPAMI)*, 2019. (Cited on page 53.)

Pan, Bowen; Panda, Rameswar; Jiang, Yifan; Wang, Zhangyang; Feris, Rogerio, and Oliva, Aude. IA-RED^2: Interpretability-aware redundancy reduction for vision transformers. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2021a. (Cited on pages 32 and 111.)

Pan, Zizheng; Zhuang, Bohan; Liu, Jing; He, Haoyu, and Cai, Jianfei. Scalable vision transformers with hierarchical pooling. In *IEEE/CVF International Conference on Computer Vision (ICCV)*, 2021b. (Cited on pages 31, 92, 107 and 111.)

Perronnin, Florent; Sánchez, Jorge, and Mensink, Thomas. Improving the fisher kernel for large-scale image classification. In *European Conference on Computer Vision (ECCV)*, pages 143–156, 2010. (Cited on page 23.)

Piergiovanni, AJ and Ryoo, Michael. Avid dataset: Anonymized videos from diverse countries. *Advances in Neural Information Processing Systems (NeurIPS)*, 2020. (Cited on page 53.)

Poppe, Ronald. A survey on vision-based human action recognition. *Image and vision computing*, 28(6):976–990, 2010. (Cited on page 1.)

Pourheydari, Saber; Bahrami, Emad; Fayyaz, Mohsen; Francesca, Gianpiero; Noroozi, Mehdi, and Gall, Juergen. Taylorswiftnet: Taylor driven temporal modeling for swift future frame prediction. In *British Machine Vision Conference (BMVC)*, 2022. (Cited on page 155.)

Qiu, Zhaofan; Yao, Ting, and Mei, Tao. Learning spatio-temporal representation with pseudo-3d residual networks. In *IEEE International Conference on Computer Vision (ICCV)*, 2017. (Cited on page 46.)

Qiu, Zhaofan; Yao, Ting; Ngo, Chong-Wah; Tian, Xinmei, and Mei, Tao. Learning spatio-temporal representation with local and global diffusion. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019. (Cited on page 113.)

Radosavovic, Ilija; Kosaraju, Raj Prateek; Girshick, Ross; He, Kaiming, and Dollár, Piotr. Designing network design spaces. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020. (Cited on page 111.)

Rao, Yongming; Zhao, Wenliang; Liu, Benlin; Lu, Jiwen; Zhou, Jie, and Hsieh, Cho-Jui. Dynamicvit: Efficient vision transformers with dynamic token sparsification. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2021a. (Cited on pages 3, 31, 90, 92, 95, 107, 108 and 111.)

Rao, Yongming; Zhao, Wenliang; Zhu, Zheng; Lu, Jiwen, and Zhou, Jie. Global filter networks for image classification. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2021b. (Cited on page 30.)

Ray, Jamie; Wang, Heng; Tran, Du; Wang, Yufei; Feiszli, Matt; Torresani, Lorenzo, and Paluri, Manohar. Scenes-objects-actions: A multi-task, multi-label video dataset. In *European Conference on Computer Vision and Pattern Recognition (ECCV)*, 2018. (Cited on pages 52 and 53.)

Richard, Alexander; Kuehne, Hilde, and Gall, Juergen. Weakly supervised action learning with rnn based fine-to-coarse modeling. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017. (Cited on pages 4, 5, 26, 28, 33, 116, 130, 131, 141, 146 and 147.)

Richard, Alexander; Kuehne, Hilde, and Gall, Juergen. Action sets: Weakly supervised action segmentation without ordering constraints. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018a. (Cited on pages xvi, 4, 26, 28, 32, 116, 117, 124, 126 and 127.)

Richard, Alexander; Kuehne, Hilde; Iqbal, Ahsan, and Gall, Juergen. Neuralnetwork-viterbi: A framework for weakly supervised video learning. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018b. (Cited on pages 4, 5, 26, 28, 33, 116, 130, 131, 140, 141, 142, 144 and 146.)

Rohrbach, Marcus; Amin, Sikandar; Andriluka, Mykhaylo, and Schiele, Bernt. A database for fine grained activity detection of cooking activities. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012. (Cited on page 32.)

Roy, Aurko; Saffar, Mohammad; Vaswani, Ashish, and Grangier, David. Efficient content-based sparse attention with routing transformers. In *Transactions of the Association for Computational Linguistics*, volume 9, pages 53–68, 2021. (Cited on page 31.)

Rumelhart, David E; Hinton, Geoffrey E, and Williams, Ronald J. Learning representations by back-propagating errors. *nature*, 323(6088):533–536, 1986. (Cited on page 13.)

Ryoo, Michael S.; Piergiovanni, AJ; Arnab, Anurag; Dehghani, Mostafa, and Angelova, Anelia. Tokenlearner: What can 8 learned tokens do for images and videos? *arXiv preprint arXiv:2106.11297*, 2021. (Cited on pages 32, 45, 62, 83, 84, 107, 112 and 113.)

Sánchez, Jorge; Perronnin, Florent; Mensink, Thomas, and Verbeek, Jakob. Image classification with the fisher vector: Theory and practice. *International journal of computer vision (IJCV)*, pages 222–245, 2013. (Cited on page 23.)

Scovanner, Paul; Ali, Saad, and Shah, Mubarak. A 3-dimensional sift descriptor and its application to action recognition. In *ACM'MM*, 2007. (Cited on page 29.)

Sharif Razavian, Ali; Azizpour, Hossein; Sullivan, Josephine, and Carlsson, Stefan. Cnn features off-the-shelf: An astounding baseline for recognition. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, 2014. (Cited on page 24.)

Sharma, Vivek; Tapaswi, Makarand; Sarfraz, M Saquib, and Stiefelhagen, Rainer. Self-supervised learning of face representations for video face clustering. In *International Conference on Automatic Face and Gesture Recognition*, 2019. (Cited on page 63.)

Simonyan, Karen and Zisserman, Andrew. Two-stream convolutional networks for action recognition in videos. In *Advances in neural information processing systems (NeuRIPS)*, 2014. (Cited on pages 1, 24, 29, 46 and 62.)

Simonyan, Karen and Zisserman, Andrew. Very deep convolutional networks for large-scale image recognition. In *International Conference on Learning Representations (ICLR)*, 2015. (Cited on pages 18, 74 and 90.)

Singh, Bharat; Marks, Tim K.; Jones, Michael; Tuzel, Oncel, and Shao, Ming. A multi-stream bi-directional recurrent neural network for fine-grained action detection. In *IEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. (Cited on page 134.)

Smaira, Lucas; Carreira, João; Noland, Eric; Clancy, Ellen; Wu, Amy, and Zisserman, Andrew. A short note about kinetics-700-2020. In *arXiv preprint arXiv:2010.10864*, 2020. (Cited on page 53.)

Soomro, Khurram; Zamir, Amir Roshan, and Shah, Mubarak. Ucf101: A dataset of 101 human actions classes from videos in the wild. *arXiv preprint arXiv:1212.0402*, 2012. (Cited on pages 27, 36, 41, 42, 51, 53, 73 and 80.)

Souri, Yaser; Richard, Alexander; Minciullo, Luca, and Gall, Juergen. On Evaluating Weakly Supervised Action Segmentation Methods. In *arXiv*, 2020. (Cited on page 141.)

Souri, Yaser; Fayyaz, Mohsen; Minciullo, Luca; Francesca, Gianpiero, and Gall, Juergen. Fast Weakly Supervised Action Segmentation Using Mutual Consistency. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 2022. (Cited on pages xvii, 9, 24, 116 and 146.)

Spriggs, Ekaterina H; De La Torre, Fernando, and Hebert, Martial. Temporal segmentation and activity classification from first-person sensing. In *CVPR Workshops*, 2009. (Cited on page 32.)

Stroud, Jonathan C; Ross, David A; Sun, Chen; Deng, Jia, and Sukthankar, Rahul. D3d: Distilled 3d networks for video action recognition. *arXiv:1812.08249*, 2018. (Cited on page 30.)

Sukhbaatar, Sainbayar; Grave, Edouard; Bojanowski, Piotr, and Joulin, Armand. Adaptive attention span in transformers. In *ACL*, 2019. (Cited on page 31.)

Touvron, Hugo; Cord, Matthieu; Douze, Matthijs; Massa, Francisco; Sablayrolles, Alexandre, and Jegou, Herve. Training data-efficient image transformers and distillation through attention. In *International Conference on Machine Learning (ICML)*, 2021. (Cited on pages x, xv, 3, 30, 90, 92, 95, 96, 99, 100, 103, 105, 107, 108 and 111.)

Tran, Du; Bourdev, Lubomir; Fergus, Rob; Torresani, Lorenzo, and Paluri, Manohar. Learning spatiotemporal features with 3d convolutional networks. In *IEEE/CVF International Conference on Computer Vision (ICCV)*, 2015. (Cited on pages 2, 19, 30, 36, 37, 46, 62, 68, 84 and 90.)

Tran, Du; Ray, Jamie; Shou, Zheng; Chang, Shih-Fu, and Paluri, Manohar. Convnet architecture search for spatiotemporal feature learning. *arXiv:1708.05038*, 2017. (Cited on pages 37, 40, 46, 57, 68, 90 and 153.)

Tran, Du; Wang, Heng; Torresani, Lorenzo; Ray, Jamie; LeCun, Yann, and Paluri, Manohar. A closer look at spatiotemporal convolutions for action recognition. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018a. (Cited on pages 30 and 31.)

Tran, Du; Wang, Heng; Torresani, Lorenzo; Ray, Jamie; LeCun, Yann, and Paluri, Manohar. A closer look at spatiotemporal convolutions for action recognition. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018b. (Cited on pages 19, 62, 70, 71, 73, 78, 83 and 84.)

Tran, Du; Wang, Heng; Torresani, Lorenzo, and Feiszli, Matt. Video classification with channel-separated convolutional networks. *IEEE/CVF International Conference on Computer Vision (ICCV)*, 2019. (Cited on pages 62 and 112.)

van den Oord, Aäron; Dieleman, Sander; Zen, Heiga; Simonyan, Karen; Vinyals, Oriol; Graves, Alex; Kalchbrenner, Nal; Senior, Andrew W., and Kavukcuoglu, Koray. Wavenet: A generative model for raw audio. In *SSW*, 2016. (Cited on pages 19, 118 and 134.)

Vaswani, Ashish; Shazeer, Noam; Parmar, Niki; Uszkoreit, Jakob; Jones, Llion; Gomez, Aidan N; Kaiser, Ł ukasz, and Polosukhin, Illia. Attention is all you need. In *Advances in Neural Information Processing Systems (NeuRIPS)*, 2017. (Cited on pages 20, 30 and 93.)

Wang, Bairui; Ma, Lin; Zhang, Wei; Jiang, Wenhao; Wang, Jingwen, and Liu, Wei. Controllable video captioning with pos sequence guidance based on gated fusion network. In *IEEE/CVF International Conference on Computer Vision (ICCV)*, 2019. (Cited on pages xiv, 62 and 63.)

Wang, Heng and Schmid, Cordelia. Action recognition with improved trajectories. In *IEEE/CVF International Conference on Computer Vision (ICCV)*, 2013. (Cited on pages 22, 23, 29 and 46.)

Wang, Heng; Kläser, Alexander; Schmid, Cordelia, and Liu, Cheng-Lin. Dense trajectories and motion boundary descriptors for action recognition. *International Journal of Computer Vision (IJCV)*, 2013. (Cited on pages vii and 23.)

Wang, Heng; Tran, Du; Torresani, Lorenzo, and Feiszli, Matt. Video modeling with correlation networks. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020a. (Cited on page 112.)

Wang, Junbo; Wang, Wei; Huang, Yan; Wang, Liang, and Tan, Tieniu. M3: Multimodal memory modelling for video captioning. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018a. (Cited on page 63.)

Wang, Limin; Qiao, Yu, and Tang, Xiaoou. Action recognition with trajectory-pooled deep-convolutional descriptors. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015. (Cited on page 46.)

Wang, Limin; Xiong, Yuanjun; Wang, Zhe; Qiao, Yu; Lin, Dahua; Tang, Xiaoou, and Van Gool, Luc. Temporal segment networks: Towards good practices for deep action recognition. In *European Conference on Computer Vision (ECCV)*, 2016. (Cited on pages 29, 46 and 62.)

Wang, Limin; Li, Wei; Li, Wen, and Van Gool, Luc. Appearance-and-relation networks for video classification. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018b. (Cited on page 62.)

Wang, Wenhai; Xie, Enze; Li, Xiang; Fan, Deng-Ping; Song, Kaitao; Liang, Ding; Lu, Tong; Luo, Ping, and Shao, Ling. Pyramid vision transformer: A versatile backbone for dense prediction without convolutions. In *IEEE/CVF International Conference on Computer Vision (ICCV)*, 2021. (Cited on page 111.)

Wang, Xiaofang; Xiong, Xuehan; Neumann, Maxim; Piergiovanni, A. J.; Ryoo, Michael S.; Angelova, Anelia; Kitani, Kris M., and Hua, Wei. Attentionnas: Spatiotemporal attention cell search for video classification. In *European Conference on Computer Vision (ECCV)*, 2020b. (Cited on page 113.)

Wang, Xiaolong; Girshick, Ross; Gupta, Abhinav, and He, Kaiming. Non-local neural networks. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018c. (Cited on pages 30, 74 and 112.)

Wang, Zhenzhi; Gao, Ziteng; Wang, Limin; Li, Zhifeng, and Wu, Gangshan. Boundary-aware cascade networks for temporal action segmentation. In *European Conference on Computer Vision (ECCV)*, pages 34–51, 2020c. (Cited on page 32.)

Willems, Geert; Tuytelaars, Tinne, and Van Gool, Luc. An efficient dense and scale-invariant spatio-temporal interest point detector. In *European Conference on Computer Vision (ECCV)*, 2008. (Cited on page 29.)

Wu, Chao-Yuan; Girshick, Ross; He, Kaiming; Feichtenhofer, Christoph, and Krahenbuhl, Philipp. A multigrid method for efficiently training video models. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020. (Cited on pages 79, 80 and 84.)

Wu, Haiping; Xiao, Bin; Codella, Noel; Liu, Mengchen; Dai, Xiyang; Yuan, Lu, and Zhang, Lei. Cvt: Introducing convolutions to vision transformers. In *IEEE/CVF International Conference on Computer Vision (ICCV)*, 2021. (Cited on pages xv, 3, 90, 92, 107, 109 and 111.)

Wu, Yuxin and He, Kaiming. Group normalization. In *ECCV*, 2018. (Cited on page 141.)

Wu, Zuxuan; Xiong, Caiming; Jiang, Yu-Gang, and Davis, Larry S. Liteeval: A coarse-to-fine framework for resource efficient video recognition. In *Advances in neural information processing systems (NeuRIPS)*, 2019a. (Cited on page 31.)

Wu, Zuxuan; Xiong, Caiming; Ma, Chih-Yao; Socher, Richard, and Davis, Larry S. Adaframe: Adaptive frame selection for fast video recognition. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019b. (Cited on page 31.)

Xie, Saining; Sun, Chen; Huang, Jonathan; Tu, Zhuowen, and Murphy, Kevin. Rethinking spatiotemporal feature learning for video understanding. In *European Conference on Computer Vision (ECCV)*, 2018a. (Cited on pages 30 and 31.)

Xie, Saining; Sun, Chen; Huang, Jonathan; Tu, Zhuowen, and Murphy, Kevin. Rethinking spatiotemporal feature learning: Speed-accuracy trade-offs in video classification. In *European Conference on Computer Vision (ECCV)*, 2018b. (Cited on pages xiv, 73, 78 and 83.)

Xu, Jun; Mei, Tao; Yao, Ting, and Rui, Yong. Msr-vtt: A large video description dataset for bridging video and language. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. (Cited on pages 62 and 63.)

Xu, Weijian; Xu, Yifan; Chang, Tyler, and Tu, Zhuowen. Co-scale conv-attentional image transformers. In *arXiv preprint arXiv:2104.06399*, 2021. (Cited on page 111.)

Yan, Shen; Xiong, Xuehan; Arnab, Anurag; Lu, Zhichao; Zhang, Mi; Sun, Chen, and Schmid, Cordelia. Multiview transformers for video recognition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3333–3343, June 2022. (Cited on page 30.)

Yi, Fangqiu; Wen, Hongyu, and Jiang, Tingting. Asformer: Transformer for action segmentation. In *British Machine Vision Conference (BMVC)*, 2021. (Cited on pages 24 and 32.)

Yu, Xumin; Rao, Yongming; Wang, Ziyi; Liu, Zuyan; Lu, Jiwen, and Zhou, Jie. Pointr: Diverse point cloud completion with geometry-aware transformers. In *IEEE/CVF International Conference on Computer Vision (ICCV)*, 2021. (Cited on page 30.)

Yuan, Li; Chen, Yunpeng; Wang, Tao; Yu, Weihao; Shi, Yujun; Jiang, Zihang; Tay, Francis EH; Feng, Jiashi, and Yan, Shuicheng. Tokens-to-token vit: Training vision transformers from scratch on imagenet. In *arXiv preprint arXiv:2101.11986*, 2021. (Cited on page 111.)

Yue, Xiaoyu; Sun, Shuyang; Kuang, Zhanghui; Wei, Meng; Torr, Philip; Zhang, Wayne, and Lin, Dahua. Vision transformer with progressive sampling. In *IEEE/CVF International Conference on Computer Vision (ICCV)*, 2021. (Cited on pages xv, 32, 92, 107, 109 and 111.)

Yue-Hei Ng, Joe; Hausknecht, Matthew; Vijayanarasimhan, Sudheendra; Vinyals, Oriol; Monga, Rajat, and Toderici, George. Beyond short snippets: Deep networks for video classification. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015. (Cited on pages 2 and 36.)

Zhao, Hang; Yan, Zhicheng; Torresani, Lorenzo, and Torralba, Antonio. Hacs: Human action clips and segments dataset for recognition and temporal localization. *arXiv:1712.09374*, 2019. (Cited on pages 51 and 53.)

Zhao, Hengshuang; Jiang, Li; Jia, Jiaya; Torr, Philip, and Koltun, Vladlen. Point transformer. In *IEEE/CVF International Conference on Computer Vision (ICCV)*, 2021. (Cited on page 30.)

Zhao, Yue; Xiong, Yuanjun; Wang, Limin; Wu, Zhirong; Tang, Xiaoou, and Lin, Dahua. Temporal action detection with structured segment networks. In *IEEE/CVF International Conference on Computer Vision (ICCV)*, 2017. (Cited on page 32.)

Zheng, Sixiao; Lu, Jiachen; Zhao, Hengshuang; Zhu, Xiatian; Luo, Zekun; Wang, Yabiao; Fu, Yanwei; Feng, Jianfeng; Xiang, Tao; Torr, Philip H.S., and Zhang, Li. Rethinking semantic segmentation from a sequence-to-sequence perspective with transformers. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021. (Cited on page 30.)

Zhou, Daquan; Kang, Bingyi; Jin, Xiaojie; Yang, Linjie; Lian, Xiaochen; Jiang, Zihang; Hou, Qibin, and Feng, Jiashi. Deepvit: Towards deeper vision transformer. *arXiv preprint arXiv:2103.11886*, 2021. (Cited on page 30.)

Zhu, Linchao; Tran, Du; Sevilla-Lara, Laura; Yang, Yi; Feiszli, Matt, and Wang, Heng. Faster recurrent networks for efficient video classification. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 13098–13105, 2020. (Cited on pages 2, 31, 69, 70, 79 and 83.)

Zolfaghari, Mohammadreza; Singh, Kamaljeet, and Brox, Thomas. Eco: Efficient convolutional network for online video understanding. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 695–712, 2018. (Cited on page 30.)