Maximizing Information from User-Clicks for Efficient Instance Segmentation in Images

Dissertation

zur

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

der

Mathematisch-Naturwissenschaftlichen Fakultät

der

Rheinischen Friedrich–Wilhelms–Universität Bonn

vorgelegt von

Soumajit MAJUMDER

aus

Rampurhat, Indien

Bonn 2022

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

1. Gutachterin und Betreuerin: Angela Yao
2. Gutachter: Jürgen Gall
Tag der Promotion: 25.01.2022
Erscheinungsjahr: 2022

Abstract

by Soumajit Majumder

for the degree of

Doctor rerum naturalium

Interactive instance segmentation allows users to select and obtain accurate pixel-level masks for objects of interest by providing inputs such as clicks, scribbles, or bounding boxes. It has always been a problem of interest in computer vision research, as it addresses quality problems faced by fully automated segmentation methods. The segmented results are helpful for downstream applications such as human-machine collaborative annotation, image/video editing, and mage-based medical diagnosis. The goal is to obtain accurate pixel-level masks for objects with minimal user input. In this dissertation, we propose several frameworks for performing interactive instance segmentation using user-provided clicks.

In interactive instance segmentation, users give feedback to refine segmentation masks iteratively. Typically, such frameworks refine false negatives and false positive regions via a succession of 'positive' and 'negative' clicks placed centrally in these regions. These user-provided 'positive' and 'negative' clicks are transformed into separate guidance maps that provide the network with necessary cues on the whereabouts of the object of interest. Most interactive frameworks incorporate these guidance maps at the image input layer. Our work proposes a novel transformation of user clicks to generate content-aware and location-aware guidance maps that leverage the hierarchical structural information present in an image. Using our guidance maps, even the most basic fully convolutional networks (FCNs) are able to outperform existing approaches that require state-of-the-art segmentation networks. Next, we propose an intuitive alternative for 'positive' and 'negative' refinement clicking by letting users click on the object boundary. We also propose a new multi-stage guidance framework for interactive segmentation. By incorporating user cues at different stages of the network, we allow user interactions to impact the final segmentation output more directly. We investigate and address challenges pertaining to user-click representation, refinement strategy, and network design in this work.

Through this dissertation, we advanced the state-of-the-art in interactive instance segmentation, proposed novel user click transformations and refinement strategies, presented new insights on the task-specialized design of such interactive frameworks. We demonstrated the effectiveness of our frameworks through comprehensive experimentation and by comparing them with existing state-of-the-art on standardized public benchmarks. We conclude this dissertation by presenting open challenges and outlining future research directions for interactive instance segmentation research.

Keywords: interactive segmentation, image segmentation, instance segmentation.

Zusammenfassung

von Soumajit Majumder

zur Erlangung des Doktorgrades

Doctor rerum naturalium

Die interaktive Instanzsegmentierung ermöglicht es Benutzern, genaue Masken auf Pixelebene für Objekte von Interesse auszuwählen und zu erhalten, indem Eingaben wie Klicks, Umrisse oder Objekt-Boxen bereitgestellt werden. Es war schon immer ein interessantes Problem in der Computer-Vision-Forschung, da es Qualitätsprobleme adressiert, die bei vollautomatischen Segmentierungsmethoden auftreten. Die segmentierten Ergebnisse sind hilfreich für nachgelagerte Anwendungen wie kollaborative Annotation zwischen Mensch und Maschine, Bild-/Videobearbeitung und bildbasierte medizinische Diagnose. Das Ziel ist es, genaue Masken auf Pixelebene für Objekte mit minimaler Benutzereingabe zu erhalten. In dieser Dissertation schlagen wir mehrere Ansätze für die Durchführung interaktiver Instanzsegmentierung mit benutzerdefinierten Klicks vor.

Bei der interaktiven Instanzsegmentierung geben die Benutzer Feedback, um die Segmentierungsmasken iterativ zu verfeinern. Typischerweise verfeinern solche Ansätze falsch-negative und falsch-positive Regionen durch eine Folge von ‘positiven’ und ‘negativen’ Klicks, die zentral in diesen Regionen platziert werden. Diese vom Benutzer bereitgestellten ‘positiven’ und ‘negativen’ Klicks werden in separate Orientierungskarten umgewandelt, die dem Netzwerk die notwendigen Hinweise auf den Verbleib des zu segmentierenden Objekts geben. Die meisten interaktiven Ansätze integrieren diese Orientierungskarten in der Bildeingabe-Schicht. Diese Arbeit schlägt eine neuartige Transformation von Benutzerklicks vor, um inhalts- und ortsbezogene Orientierungskarten zu erzeugen, die die hierarchischen Strukturinformationen eines Bildes nutzen. Mit unseren Orientierungskarten sind selbst die einfachsten Faltungsnetze in der Lage, bestehende Ansätze zu übertreffen, die hochmoderne Segmentierungsnetzwerke erfordern. Als nächstes schlagen wir eine intuitive Alternative für ‘positive’ und ‘negative’ Verfeinerungsklicks vor, indem wir den Benutzer auf die Objektgrenze klicken lassen. Außerdem schlagen wir ein neues mehrstufiges Anleitungskonzept für die interaktive Segmentierung vor. Durch die Einbeziehung von Benutzerhinweisen in verschiedenen Phasen des Netzwerks ermöglichen wir, dass Benutzerinteraktionen die endgültige Segmentierungsausgabe direkter beeinflussen. In dieser Arbeit untersuchen und adressieren wir Herausforderungen, die die Benutzer-Klick-Darstellung, die

Verfeinerungsstrategie und das Netzwerkdesign betreffen.

In dieser Arbeit haben wir den Stand der Technik bei der interaktiven Instanzsegmentierung weiterentwickelt, neuartige Benutzer-Klick-Transformationen und Verfeinerungsstrategien vorgeschlagen und neue Erkenntnisse über das aufgabenspezifische Design solcher interaktiven Ansätze präsentiert. Wir demonstrierten die Effektivität unserer Ansätze durch umfassende Experimente und durch Vergleiche mit dem bestehenden Stand der Technik auf standardisierten öffentlichen Benchmarks. Wir schließen diese Dissertation ab, indem wir offene Herausforderungen präsentieren und zukünftige Forschungsrichtungen für die interaktive Instanzsegmentierungsforschung skizzieren.

Schlagwörter: interaktive Segmentierung, Bildsegmentierung, Instanzsegmentierung.

Acknowledgements

First and foremost, I want to thank my advisor, Angela Yao, for giving me the opportunity to work under her supervision. I am incredibly grateful for her support, for her research ideas and enthusiasm, and for giving me the freedom to pursue my ideas. I learned a lot from Angela about how to structure, conduct, and present research ideas. Without her constant input and feedback, this dissertation would not have been possible.

I want to thank all my dissertation committee members for kindly accepting my request and for evaluating my work.

Next, I want to thank my friends and colleagues at the University for creating a friendly and warm work environment. I want to thank Fadime, Moritz, and Linlin for always being available for conversations and suggestions. It was a pleasure to share the office with you. I would also like to thank Sebastian, Michael, Ralf, Simone, and Reinhard Klein for helping me throughout my stay at the University.

Special thanks to Ansh and Abhinav, with whom I had the opportunity to work during their time at the NUS. I would also like to thank Sovan and Amin for making my stay in Bonn full of great experiences.

I want to thank Partha and Chandreyee for their support and encouragement throughout this journey and for being the family far from home.

Last but not least, I would like to thank my parents for their unwavering belief in me over all these years.

Contents

List of Figures

List of Tables

Nomenclature

Abbreviations

An alphabetically sorted list of abbreviations used in this dissertation:

BCE	binary cross-entropy
BN	batch normalization
CNN	convolutional neural network
DAVIS	densely annotated video segmentation
DNN	deep neural network
DL	deep learning
FC	fully connected
FCN	fully convolutional network
GCN	graph convolutional network
IoU	intersection over union
iVOS	interactive video object segmentation
MCG	multiscale combinatorial grouping
mIoU	mean intersection over union
ML	machine learning
MLP	multilayer perceptron
R-CNN	region based convolutional neural networks
ReLU	rectified linear units
RNN	recurrent neural network
SGD	stochastic gradient descent
SVM	support vector machine

Chapter 1

Introduction

1.1 Motivation

Image segmentation. Understanding and reasoning about structures in an image is a key area of computer vision research. Image segmentation is the first step in that direction (Fu and Mui, 1981). In computer vision research, image segmentation refers to the task of partitioning an image, i.e. grouping image pixels, into regions that are simultaneously compact and expressive (Ballard and Brown, 1982). It is one of the most fundamental and challenging problems in computer vision. Image segmentation enables key technologies like autonomous driving (Cordts et al., 2016), medical image analysis (Rosenfeld, 1976; Stalling and Hege, 1996; Wang et al., 2018b), image search engines (Wan et al., 2014), smart phone photography (Chen et al., 2018a). It still remains an active area of research (Hao et al., 2020; Hafiz and Bhat, 2020), with literature dating back to over half a century (Doyle, 1962; Wacker and Landgrebe, 1970). Early image segmentation algorithms include threshold selection techniques (Doyle, 1962; Prewitt and Mendelsohn, 1966; Weszka, 1978), clustering (Wacker and Landgrebe, 1970), edge detection (Davis, 1975), region merging techniques (Rosenfeld, 1976; Fu and Mui, 1981; Jain and Dubes, 1988) among others. Image segmentation is a critical component in an image processing pipeline; errors in segmentation will negatively impact feature extraction, classification, and interpretation (Jain and Dubes, 1988). Early segmentation algorithms assigned class labels to pixel based on low-level features. The lack of semantic understanding meant that masks generated were often erroneous and undesirable. Interactive instance segmentation algorithms became popular as a means to alleviate the imperfections inherent to these early segmentation methods (Kass et al., 1988). Interactive instance segmentation allows users to select objects of interest down to the pixel level by providing inputs such as clicks (Boykov and Jolly, 2001), scribbles (Li et al., 2004; Bai and Wu, 2014), or bounding boxes (Rother et al., 2004; Maninis et al., 2018).

Interactive image segmentation. GrabCut (Rother et al., 2004) is a pioneering example of interactive segmentation. It segments object instances initialized with a user-marked bounding box by iteratively updating a color-based Gaussian mixture model (GMMs). GrabCut achieves good performance when the object and background color distributions are well-separable. However, such methods try to estimate the object and background distributions from low-level features

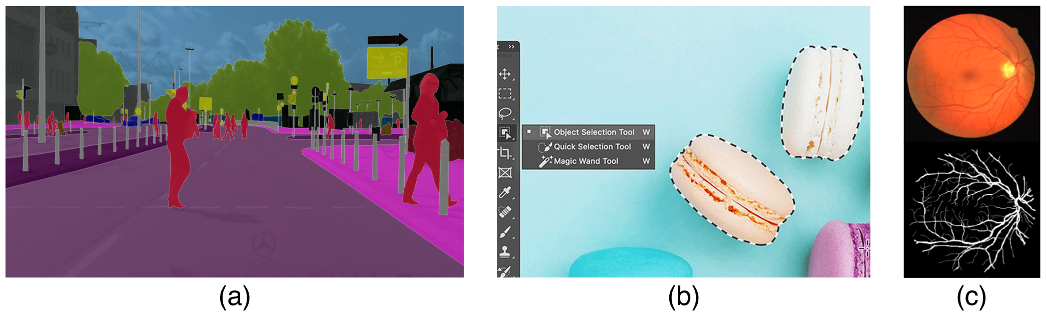


Figure 1.1: Image segmentation drives several crucial technologies. (a) Image from Cityscapes dataset (Cordts et al., 2016). In autonomous driving, it is crucial to locate and outline all pedestrians, cars, traffic signs, and other static and dynamic objects. With recent advances in image segmentation, self-driving cars equipped with state-of-the-art computer vision models can localize multiple objects and safely navigate challenging urban pathways. (b) Commercial software such as Photoshop also benefit from image segmentation methods. With newly launched features such as the object selection tool, Photoshop CC users can obtain a precise selection of objects with few clicks instead of tediously tracing pixels.[1](c) Image from DRIVE dataset (Niemeijer et al., 2004). Retinal vessel segmentation is critical for medical image analysis as it helps early diagnosis of ophthalmic disorders. Automated segmentation of such vessels allows the development of computer-aided diagnosis systems.

such as color (Rother et al., 2004) and texture. Such low-level features are unfortunately ineffective in several instances, e.g. in images with similar foreground and background appearances (Kass et al., 1988; Mortensen and Barrett, 1995; Boykov and Jolly, 2001; Rother et al., 2004), intricate textures (Mortensen and Barrett, 1995), and poor illumination. Often, this results in low-level feature based incoherent segmentation of an object. In order to obtain sufficiently accurate segmentation masks, all these methods require substantially more user interactions thereby increasing the burden on the user (Xu et al., 2016). Despite their shortcomings, early interactive instance segmentation approaches remained popular with some variants becoming part of commercially successful software packages.

Deep learning and image segmentation. Riding on the advances in CNN architectures (Simonyan and Zisserman, 2014; He et al., 2016) and transfer learning (Zeiler and Fergus, 2014), computer vision systems improved significantly, especially in tasks like image segmentation, e.g. semantic segmentation (Long et al., 2015; Liang-Chieh et al., 2015; Chen et al., 2018b; Zhao et al., 2017; Chen et al., 2018c), instance segmentation (He et al., 2017; Wang et al., 2019), and panoptic

[1]https://www.theverge.com/2019/11/4/20943796/adobe-photoshop-object-selection-tool-cloud-psd-update

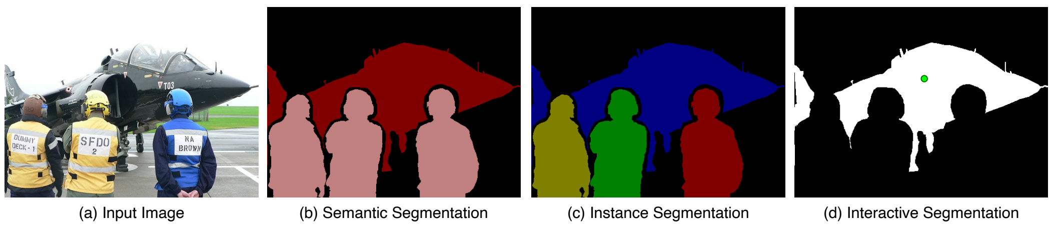

(a) Input Image
(b) Semantic Segmentation
(c) Instance Segmentation
(d) Interactive Segmentation

Figure 1.2: We can broadly divide image segmentation into three types. In semantic segmentation, all objects of the same class are marked using the same label. In instance segmentation, similar objects are assigned their own unique labels alongside the class label. In interactive segmentation, the user specifies the target object (shown using a green circle), and the corresponding segmentation mask is generated.

segmentation (Xiong et al., 2019). Fully convolutional networks (FCNs) (Long et al., 2015) is a notable example; it laid the foundation for successful semantic segmentation models which followed. Availability of high quality large-scale datasets laid the foundation for these successful segmentation models (Castrejon et al., 2017; Acuna et al., 2018; Ling et al., 2019). However, instance segmentation is often considered as one of the most expensive and difficult to annotate (Benenson et al., 2019). Manual labelling of such data is highly laborious. Annotating single objects can take up to 40 seconds per object (Castrejon et al., 2017; Acuna et al., 2018) and a full image can take as long as 1.5 hours for Cityscapes (Cordts et al., 2016).

The need for large scale annotations. Over time, segmentation models became larger and their performance more dependent on the volume of training data (Chen et al., 2018c). There arose a need in the computer vision community to annotate large-scale datasets to sustain the progress (Castrejon et al., 2017). Manual annotation on the other hand continued to remain expensive. In order to reduce the dependency on detailed annotations such as per-pixel masks, several segmentation methods also considered training in a weakly supervised setting (Dai et al., 2015; Jain and Grauman, 2016; Khoreva et al., 2017). These approaches aim to learn segmentation models from weak annotations such as image labels or bounding boxes (Dai et al., 2015; Khoreva et al., 2017). Other approaches relied on scribbles (Lin et al., 2016) or single point (Bearman et al., 2016) on each of the objects. Although these approaches demonstrate promising performance, they are not yet competitive with their fully supervised counterparts leveraging detailed annotations. Thus, the need for large-scale datasets with high quality segmentation masks persisted.

Human-machine collaborative annotation. To sustain the progress of vision models, one now could leverage the advances in human-in-the-loop interactive

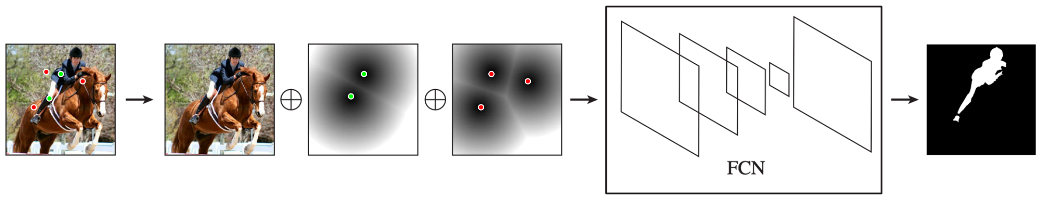


Figure 1.3: Interactive framework of Xu et al. (2016). Given an input image with an object of interest and user clicks, the positive and negative clicks (marked in green and red respectively) are separately transformed into guidance maps and concatenated (denoted as $\oplus$) with the RGB channels to compose an input pair to the FCN models. Desired output is the ground truth mask of the target instance. Image from Xu et al. (2016).

instance segmentation frameworks for the annotation of large scale datasets (Benenson et al., 2019; Andriluka et al., 2018; Maninis et al., 2018; Ling et al., 2019; Castrejon et al., 2017; Acuna et al., 2018). Indeed, deep interactive frameworks helped speed up annotation by a significant factor (Acuna et al., 2018). By working with high-level representations encoded in FCNs, the number of user interactions required to generate quality segments greatly reduced (Xu et al., 2016). Trained on existing segmentation datasets, such as Pascal VOC 2012 (Everingham et al., 2010), deep interactive frameworks showed great promise in generalizing to unseen classes during training, e.g. annotating instances from non-overlapping object classes from the MS COCO (Lin et al., 2014) dataset.

Deep learning frameworks for interactive instance segmentation. In standard automated deep instance segmentation (Vincent and Soille, 1991; Dai et al., 2016; He et al., 2017), the RGB image is given as input and segmentation masks for each object instance are predicted. If the resulting masks are unsatisfactory, they need to be manually rectified. Post-segmentation, the model has no role to play. Instead, in deep interactive instance segmentation, users can repeatedly provide cues to the segmentation network until a satisfactory mask is obtained. In the initial work of (Xu et al., 2016), user-provided clicks are converted to Euclidean distance transform maps which are concatenated with the color channels and fed as input to a FCN (Long et al., 2015) (shown in Fig. 1.3). The input layer of the FCN was modified to accommodate the "positive" and "negative" user clicks. Clicks are then added iteratively based on the errors of the previous prediction. On arrival of each new click, the Euclidean distance transform maps are updated and inference is performed. The process is repeated until a satisfactory result is obtained. This framework of (Xu et al., 2016) was widely adopted across deep interactive frameworks (Liew et al., 2017; Mahadevan et al., 2018; Li et al., 2018; Jang and Kim, 2019; Sofiiuk et al., 2020; Lin et al., 2020). Other widely adopted extensions include the use of newer FCN architectures (Mahadevan et al., 2018; Benard and Gygli, 2017) such as Deeplabv2 (Chen et al., 2018b) and Deeplabv3+ (Chen et al., 2018c) as the segmentation backbone, Gaussian user-click transformation (Mahadevan et al., 2018), and iterative training procedures (Mahadevan et al., 2018; Liew et al., 2017).

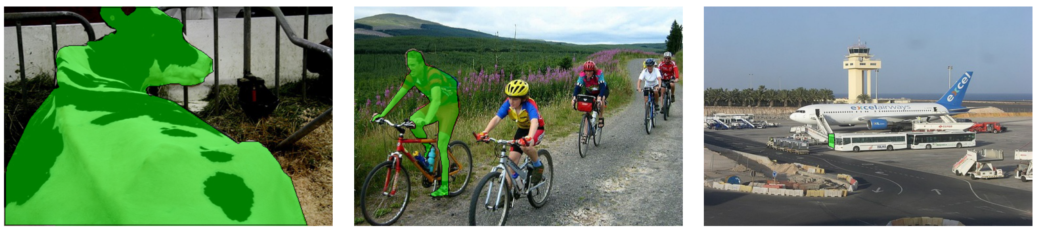
Interactive frameworks in current day. To date, interactive instance segmentation remains an active area of research (Lin et al., 2020). Other than addressing the frailties of segmentation algorithms and models, interactive frameworks form the key technology for a variety of downstream applications which desire high quality segment masks. These include image and video editing for digital image composition (Mortensen and Barrett, 1995; Rother et al., 2004; Li et al., 2004; Benard and Gygli, 2017; Li et al., 2004), image-based medical diagnosis (Wang et al., 2018a,b), autonomous driving (Cordts et al., 2016), and human-machine collaborative annotation (Andriluka et al., 2018; Benenson et al., 2019; Castrejon et al., 2017; Acuna et al., 2018; Ling et al., 2019).

Our research goal. The holy grail of interactive instance segmentation is to achieve one-click segmentation (Li et al., 2018). The user places one click on the instance of interest, and the system returns a pixel-accurate segmentation mask. Despite significant progress, advanced deep learning-based frameworks (Sofiiuk et al., 2020; Lin et al., 2020) still require multiple clicks. In this thesis, we focus on click-based interactive instance segmentation. We question design choices made in existing interactive frameworks and propose efficient interactive frameworks. We advance the current state-of-the-art approaches in interactive instance segmentation through our work as we strive towards the unified goal of one-click segmentation.

1.2 Challenges

Despite significant progress over the recent years, interactive image segmentation still remains a challenging problem in computer vision. It inherits both the challenges of image segmentation and unconstrained human-machine interaction.

Objects at multiple scales. Across the main benchmarks, such as Pascal VOC 2012 (Everingham et al., 2010), and MS COCO (Lin et al., 2014), objects can appear at multiple scales ranging from *small* (less than 32×32 pixels) to *large* (bigger than 96×96) (Fig. 1.4). Small objects, in particular, are difficult for segmentation networks to tackle; they are often ignored and classified as background pixels (Noh et al., 2015). This is an outcome of the repeated combination of down-sampling operations (max-pooling, convolution with strides) performed at consecutive layers



(a) Large objects (> 96 x 96 pixels)

(b) Medium objects (> 32 x 32 pixels and < 96 x 96 pixels)

(c) Small objects (< 32 x 32 pixels)

Figure 1.4: In natural imagery, objects can manifest themselves in different forms and sizes. Object sizes in public benchmarks, such as Pascal VOC 2012 or MS COCO, can be divided into three different categories - (a) large, (b) medium, and (c) small objects (Noh et al., 2015). When it comes to segmentation models, small objects are the most challenging to segment as they are more likely to be occluded and are even challenging for the naked human eye (as seen in (c)). The segmentation mask for the objects of interest are overlaid in green.

of the deep CNN backbones that constitute the segmentation models. The encoded feature maps have significantly lower spatial resolution and is often too coarse, e.g. a stride of 32 in FCN-32s (Long et al., 2015) and 8 in Deeplabv2 (Chen et al., 2018b).

To sidestep this limitation, segmentation networks often use skip architectures (Long et al., 2015), and *atrous* convolutions (Liang-Chieh et al., 2015; Chen et al., 2018b,c). The detailed structures in small objects are nonetheless lost. On the other hand, specifying the scale of an object have been shown to improve segmentation performance (Papandreou et al., 2015). Thus, we require approaches that can convey, either implicitly or explicitly, the scale of an object from the user interactions to the segmentation network. Small objects are particularly challenging for interactive frameworks as the user has to identify the small objects as well as refine the segmentation approaches.

Modeling user interaction. Across most interactive methods, the first click is intended for selecting the object of interest. Subsequent clicks (Xu et al., 2016; Mahadevan et al., 2018; Benard and Gygli, 2017) are then placed to refine parts of the object or correct erroneous predictions. Intuitively, the positive clicks should add portions of missing foreground, while negative clicks remove falsely segmented parts of the background. Users place clicks centrally in regions that directly corresponds with false negatives and false positives. While such succession of clicks on the false positive and false negative regions have been shown to be effective, there are two inherent limitations.

First, current refinement clicks do not specify the spatial extent of the erroneous region. Also, while positive clicks intended to remove false negative regions are always placed within the object, negative refinement clicks can occur anywhere within the image. Such succession of clicks, therefore, does not tighten the prior on the location of the object. As more refinement clicks are added, the segmentation network is unable to meaningfully leverage them. Not surprisingly, a plateauing occurs, as the average per instance mIoU tends to stagnate beyond 5-6 clicks. Therefore, we require refinement clicks with more utility which alongside refinement, also places a stronger cue on *where* the object is.

Second, depending on the order of arrival, different user clicks have distinct purposes. The first click is almost exclusively intended for selecting the object of interest. Ensuing clicks are used for refinement. This scheme of interaction, i.e. localization followed by refinement, is highly intuitive and well supported by user studies (Benenson et al., 2019). Yet, when looking at the current state-of-the-art frameworks (Xu et al., 2016; Jang and Kim, 2019; Sofiiuk et al., 2020), we find no distinction during learning between the tasks of localization versus refinement. Clicks are sampled as a group based on the ground truth masks according to some predetermined scheme (Xu et al., 2016). They are also treated equally when generating guidance maps (Benard and Gygli, 2017; Mahadevan et al., 2018; Majumder and Yao, 2019a; Xu et al., 2016) without any notion of intended purpose. Unintentionally, this can lead to sub-optimal leveraging of user clues. Therefore, we require approaches which can better leverage the user cues for their intended purpose.

User click representation. Conventionally, cross existing frameworks (Xu et al., 2016; Liew et al., 2017; Mahadevan et al., 2018; Jang and Kim, 2019; Lin et al., 2020), the input consists of the RGB image as well as *'guidance'* maps based on user-provided supervision. This supervision can come in the form of clicks, scribbles (Agustsson et al., 2019) or bounding boxes (Xu et al., 2017b; Maninis et al., 2018; Agustsson et al., 2019). In a click-based interaction setting, users give '*positive*' clicks on the object of interest and '*negative*' clicks on the background or other objects in the scene. The guidance map helps the network to focus on the object instance to segment. In the iterative setting, it helps to correct errors from previous segmentations (Benard and Gygli, 2017; Liew et al., 2017; Mahadevan et al., 2018; Xu et al., 2016).

Following the user-provided clicks, guidance maps are generated via fixed rules and are not visible to the end user. Only the image, intermediate, and end segmentation results are visible to the interacting user. Representation of user interactions to date remains an open question in interactive segmentation research. Current methods rely on distance-based primitives, e.g. Euclidean distance maps (Xu et al.,

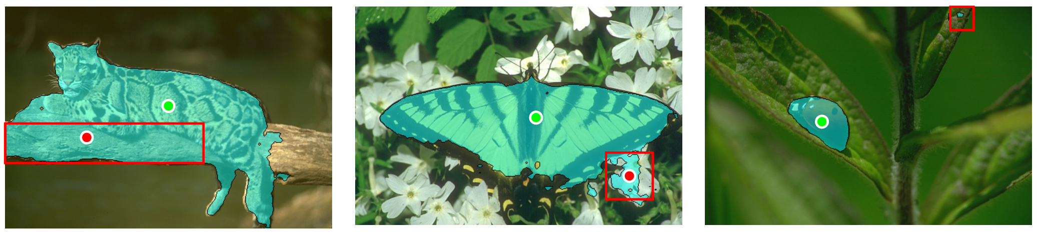


Figure 1.5: Segmentation masks generated from the networks are not always perfect. Given an initial positive click from the user (shown in green) on the object of interest, the generated segmentation masks (overlaid) shows false positive regions where the background pixels were marked as foreground. Here, the false positive error in the left image is significantly larger compared to the other images. We provide a rough visual estimate using the red bounding boxes. In order to refine the mask and remove the false positives, most interactive frameworks require the user to place a click in the center of the largest erroneous region (skipped for the right for ease of visualization). The corresponding refinement click is shown using a red circle. We note that these refinement clicks do not account for the spatial extent of the error which can vary to a large extent across different instances.

2016), Gaussians (Mahadevan et al., 2018; Benard and Gygli, 2017), which disregard the basic structures present in a scene, e.g. color and texture consistency. These user click representations lack basic image content awareness. Thus, we need user click encodings which are able to leverage the low-level and high-level structures present in the image.

1.3 Contribution

While the last few years saw a pronounced increase in the size of models, the demand for data to date remains a bottleneck (Benenson et al., 2019). The cost of manual annotation is very high; this drives the need for methodologies enabling faster and efficient scale-up of human annotations. Consequently, research activity in interactive segmentation has seen a significant uptick.

In this dissertation, we addressed different facets of the deep interactive segmentation problem pertaining to framerwork design. Our proposed interactive frameworks presented in this dissertation have been presented in conferences and published in proceedings (Majumder and Yao, 2019a,b; Majumder et al., 2020b,a).

- Soumajit Majumder, and Angela Yao. "Content-aware multi-level guidance for interactive instance segmentation." CVPR, 2019.

- Soumajit Majumder, and Angela Yao. "Localized Interactive Instance Segmentation." GCPR, 2019.
- Soumajit Majumder, Abhinav Rai, Anshul Khurana, and Angela Yao. "Two-in-One Refinement for Interactive Segmentation." BMVC, 2020.
- Soumajit Majumder, Anshul Khurana, Abhinav Rai, and Angela Yao. "Multi-Stage Fusion for One-Click Segmentation." GCPR, 2020.

Efficient encoding of user interaction. In popular interactive image segmentation, user feedback is typically given as pixel clicks (Hu et al., 2019; Li et al., 2018; Liew et al., 2017; Mahadevan et al., 2018; Maninis et al., 2018; Xu et al., 2016). To leverage these user inputs, clicks are transformed into guidance maps via simplistic primitives such as Euclidean (Hu et al., 2019; Li et al., 2018; Xu et al., 2016) or Gaussian distance maps (Benard and Gygli, 2017; Mahadevan et al., 2018; Maninis et al., 2018). These transformations disregard the basic image consistencies present in the scene, such as colour, local contours, and textures. This of course also precludes even more sophisticated structures such as object hypotheses, all of which can be determined in an unsupervised way.

Another drawback of these structure-agnostic Euclidean and Guidance-based guidance maps is that they do not account for the scale of the object during interaction. These distance maps are primarily used for localizing the user clicks. However, object scale has a direct impact on the network performance when it comes to image segmentation (Noh et al., 2015).

Our motivation is to maximize the information that can be harnessed from user-provided clicks and generate more meaningful user click transformations. As our first contribution, we propose an effective transformation of user clicks incorporating low-level cues such as color and texture, based on superpixels (Yao et al., 2015; Achanta et al., 2012), to more high-level information such as class-independent object hypotheses (Pont-Tuset et al., 2017; Maninis et al., 2016). For our second contribution, we proposed an algorithm which enables us to estimate the object scale based on the user-provided clicks and refines these guidance maps accordingly. Our work is the first work to investigate the impact of guidance map generation and the incorporation for the task of interactive instance segmentation.

Localized interaction. Interactive segmentation is a localized task; a user interested in recovering or *cutting* an object instance from the scene would focus more on delineating the object of interest from it's nearby background pixels. However, in current interactive instance segmentation works, the user is granted a free hand when providing clicks to segment an object; clicks are allowed on background pixels and other object instances far apart from the target object (Xu et al., 2016; Liew et al., 2017). Directing user clicks to specify the location may seem like an obvious way for interaction but few works on interactive segmentation have done so to date. Instead, existing frameworks impose hard constraints by directly cropping out the bounding boxes derived from user-given inputs (Maninis et al., 2018) or class-dependent object detections (Xu et al., 2017b).

Our interest is to direct and leverage user clicks to *weakly* constrain the area of interest for interactive segmentation. Limiting the spatial extent offers advantages for both for the network and the user, i.e. it tells the network which area to focus on for learning and also gives some indication of object scale; it also directs the user clicks to ambiguous locations which will most benefit from guidance.

As our contribution, we propose a new transformation scheme for the user-provided clicks. This provides a weak localization prior on the object of interest all the while being consistent with low-level image structures such as edges, textures, etc. Additionally, we generate this prior in a class-agnostic fashion. Unlike (Xu et al., 2017b), our proposed approach does not rely on class-specific bounding box detections. As more and more clicks arrive, this proposed transformation gradually refines the localization prior.

Two-in-one refinement. Across most interactive frameworks (Xu et al., 2016; Mahadevan et al., 2018; Liew et al., 2017; Hu et al., 2019; Majumder and Yao, 2019a; Liew et al., 2017; Jang and Kim, 2019) including our preceding works, the first positive click is reserved for selecting the object of interest. Refinement of the generated mask is done via a successive placement of positive and negative clicks. Positive clicks add portions of missing foreground, while negative clicks remove falsely segmented parts of the background (Xu et al., 2016; Liew et al., 2019; Li et al., 2018; Liew et al., 2017).

While such clicking strategy for refinement have been proven to be successful, to distinguish between the two forms of refinement, interactive frameworks require separate input encodings, so there are always at least two guidance maps (Xu et al., 2016; Liew et al., 2019; Li et al., 2018; Liew et al., 2017). More importantly, users are expected to identify the largest erroneous region and then place clicks centrally in those false negative and false positive regions. Boundary clicks, on the other hand, are more intuitive and easier to mark. In addition, boundary clicks have more utility, as they provide a much stronger cue on the object. As more and more boundary refinement clicks become available, the instance gets explicitly encircled with refinement clicks.

As our first contribution, instead of conventional positive and negative clicks to fix false negatives and false positives, we ask users to click on object boundaries in the vicinity of errors. Additionally, we observe that current frameworks treat all positive clicks indiscriminately. As our next contribution, we propose task-specialized framework where dedicated networks are applied to select and refine the object. As our final contribution, we propose a boundary-aware loss function which in unison with our boundary clicks provide a strong cue for interactive segmentation.

Multi-stage fusion of user interaction. The holy grail of interactive instance segmentation is to achieve one-click segmentation. The user places one click on the object of interest, and the system returns an accurate mask. This is particularly desired and crucial for downstream applications such as casual photography (Chen et al., 2018a). Such use-cases has a strong focus on maximizing the mean intersection over union (mIoU) with a single click. Despite significant progress, advanced deep learning-based frameworks still require multiple clicks. We observe that existing approaches particularly fare poorly with only one or two clicks.

To make the most of the first (few) click(s), we hypothesize that user cues' guidance should be fused into the network at multiple locations rather than via early fusion. Similar to gradients vanishing as they reach the initial layers during back-propagation, user-input cues also diminish as it progresses through the network during a forward pass. This effect is further accentuated by the many layers of the deep CNNs (Hu et al., 2019). A late fusion would allow the user interaction to have a direct and more pronounced effect on the final segmentation mask. The attempts in previous works (Hu et al., 2019; Rakelly et al., 2018) to have late-fusion of user interaction resulted in extremely parameter heavy network designs with instances requiring additional 100 million parameters (Rakelly et al., 2018). As our contribution, we propose an light-weight and easier-to-train interactive framework to perform fusion of user interactions at multiple stages of the network.

Summary of our contributions. The challenge of acquiring clicks is a recurring theme across all click-based segmentation methods. Intuitively, lower clicks are desirable for several downstream applications such as mobile photography, commercial editing software, and annotation frameworks. As such, interactive frameworks are benchmarked using the number of clicks required to reach the desired segmentation quality. In this dissertation, we propose multiple approaches for click-efficient interactive segmentation. With our proposed user-click encodings, refinement schemes, and newer network designs, we are able to attain state-of-the-art performance on several challenging public benchmarks. Finally, our work lays the foundation for task-specialized (*select* and *refine*) interactive frameworks and network designs that guarantees user clicks to have more impact on the segmentation outcome.

1.4 Organization

In this dissertation, we develop new frameworks and algorithms for performing interactive image segmentation as we made progress towards reducing annotation effort and achieving the collective goal of one-click segmentation. The thesis is organized as follows:

- In **Chapter 2**, we perform a literature review of existing interactive frameworks, both classical and deep learning based approaches. We discuss the relative advantages and shortcomings of the individual frameworks.
- In **Chapter 3**, we present an overview of the machine learning (ML) and deep learning (DL) components that constitute our interactive instance segmentation frameworks.
- In **Chapter 4**, we present our algorithm for content-aware user click representation and demonstrate how such click encoding can positively impact deep learning frameworks to reduce annotation effort. The content of this chapter corresponds to our CVPR 2019 publication "Content-aware multi-level guidance for interactive instance segmentation" (Majumder and Yao, 2019a).
- In **Chapter 5**, we introduce our localized interactive image segmentation framework. Different to existing methods, we constrain users to place clicks in the vicinity of the object. Using such a strategy allows us to impose a weak location prior which manifests itself in improved accuracy and lowered annotation effort. The content of this chapter were presented in our GCPR 2019 work "Localized Interactive Instance Segmentation" (Majumder and Yao, 2019b).
- In **Chapter 6**, we introduce our two-in-one refinement framework which simplifies user corrective clicks to be of a single type as opposed to existing frameworks requiring two types of clicking. The content of this chapter corresponds to our BMVC 2020 publication "Two-in-One Refinement for Interactive Segmentation" (Majumder et al., 2020b).
- In **Chapter 7**, we propose a multi-stage fusion architecture for light-weight yet effect incorporation of user click encoding at different layers of the segmentation network. The chapter contains our findings reported in our GCPR 2020 publication "Multi-Stage Fusion for One-Click Segmentation" (Majumder et al., 2020a).
- In **Chapter 8**, we summarize our contributions to the field of interactive image segmentation research, and we conclude this dissertation by outlining the remaining challenges and how the field will shape up in the near future.

Chapter 2

Related Works

Over the recent years, automated semantic and instance segmentation algorithms have become very effective in object delineation. However, several downstream applications like image and video editing tools (Benard and Gygli, 2017) and medical image analysis (Wang et al., 2018a; Grady et al., 2005) require exact pixel-level masks. Semantic segmentation models also benefit from training on high quality annotations (Khoreva et al., 2017). Erroneous masks generated from automated segmentation approaches are hence undesirable for such applications. Human beings, on the other hand, can effortlessly understand the semantics of the scene, and identify the foreground object(s). But it is a time-consuming and cumbersome process to manually outline the object boundaries (Kass et al., 1988) pixel by pixel. Interactive segmentation algorithms provide a solution to this by invoking the aid of a human to correct the errors. With interactive frameworks, humans can provide high-level guidance, in the form of coarse and sparse annotations (Kass et al., 1988; Boykov and Jolly, 2001; Rother et al., 2004; Xu et al., 2016), and the segmentation algorithm propagates that annotations down to the pixel level.

Early interactive instance segmentation methods include parametric active contour model (Kass et al., 1988; McInerney and Terzopoulos, 2000; Cremers et al., 2002), and intelligent scissors (Mortensen and Barrett, 1995; Barrett and Mortensen, 1997; Falcao et al., 1998). These methods primarily rely on boundary properties when performing segmentation. As a result, in the presence of low contrast and noisy boundaries, and highly textured (or un-textured) regions, these methods tend to fare poorly (Xu et al., 2016). Other methods consider both regional and boundary properties. These include methods based on graph cuts (Boykov and Jolly, 2001; Rother et al., 2004; Vezhnevets and Konouchine, 2005; Li et al., 2004), geodesics (Bai and Sapiro, 2009; Criminisi et al., 2008), and or a combination of the two (Gulshan et al., 2010; Price et al., 2010). However, the performance of such methods were limited by their lack of higher understanding of objects.

2012 is widely considered as the start of the *deep learning* revolution. The term *deep learning* (DL) refers to the branch of machine learning (ML) based on neural networks (Fukushima and Miyake, 1982) with many layers; hence the term deep. Although neural networks existed in literature, it was not until 2012 that DL-based methods started to significantly outperform other ML approaches on several challenging benchmarks (Russakovsky et al., 2015; Everingham et al., 2010;

Lin et al., 2014). Convolutional neural networks (CNNs), a class of neural network models, performed particularly well for hand-written digit classification (LeCun et al., 1998). However, CNNs went out of favor with the rise of support vector machines (SVMs) (Hearst et al., 1998) and random forests (Breiman, 2001). In 2012, they regained popularity after it substantially outperformed other competing methods, an error rate of 16.5% (Krizhevsky et al., 2012) compared to 26.1% from the 2^{nd} place method, on the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) (Deng et al., 2009; Russakovsky et al., 2015). Several factors contributed to this improvement - increase in processing ability and the decreasing cost of GPUs, better regularization strategies like Dropout (Hinton et al., 2012), and the availability of large scale training datasets with millions of manually annotated examples (Deng et al., 2009). Some well-known CNN models include AlexNet (Krizhevsky et al., 2012), GoogLeNet (Szegedy et al., 2015), VGG (Simonyan and Zisserman, 2014), and ResNet (He et al., 2016).

Research in computer vision, particularly image segmentation, greatly benefited from the advent of deep learning based methods. Deep CNNs paved the way for fully convolutional networks or FCNs (Long et al., 2015) which demonstrated exemplary performance when it comes to semantic segmentation on challenging public benchmarks such as Pascal VOC 2012 (Everingham et al., 2010) and the MS COCO (Lin et al., 2014) dataset. The work of Xu et al. (2016) was the first to propose a deep learning based interactive segmentation framework. In this seminal work, the weights of the segmentation model was initialized with the weights of contemporary state-of-the-art model on semantic segmentation, FCN-8s (Long et al., 2015). Since then, FCNs have been integral part of interactive frameworks with newer frameworks adapting the contemporary state-of-the-art semantic segmentation models, Deeplabv2 (Maninis et al., 2018), and Deeplabv3+ (Mahadevan et al., 2018; Benard and Gygli, 2017).

2.1 Classical boundary-based approaches

Snakes. Introduced in 1988, Active Contours or Snakes (Kass et al., 1988) quickly gained popularity for the task of interactive image segmentation. After being initialized with a rough boundary approximation, snakes iteratively adjust the boundary points in parallel to minimize an energy functional and achieve an optimal boundary. This energy functional is a combination of image forces, such as boundary curvature and color gradient magnitude, and external constraint forces. The image forces push the snake toward salient image features like lines, edges. The external constraint forces are responsible for putting the snake near the desired local minimum; these external forces, for example, can come from a user interface. Eventually, the snake deforms itself into conformity with the nearest

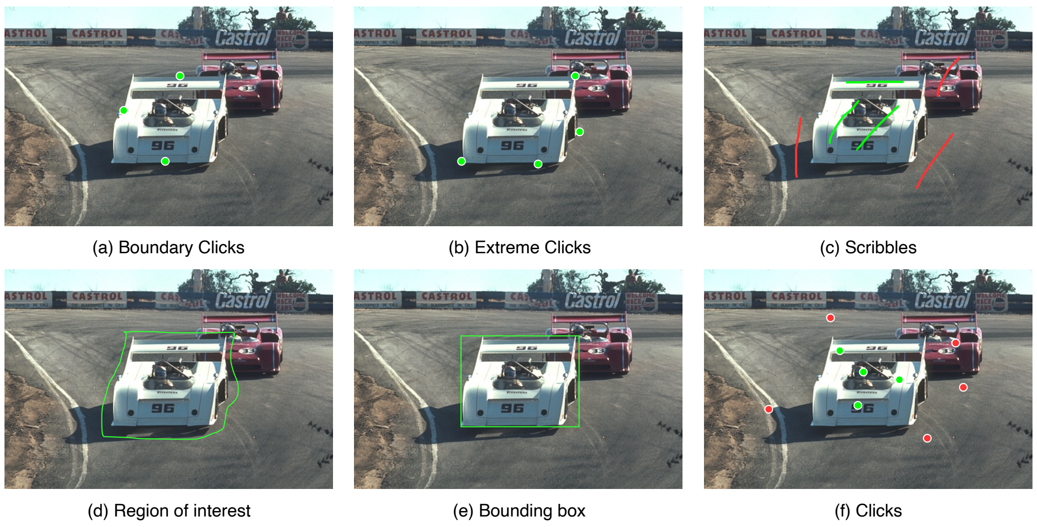


Figure 2.1: Assume a user interested in segmenting the white car. In popular interactive frameworks, a user can provide cues regarding the whereabouts of the target object (in this example, the white car) to the segmentation method via inputs such as (a) boundary clicks (Le et al., 2018), (b) extreme clicks placed on the top-right-bottom-left extremes of the object (Maninis et al., 2018), (c) scribbles (Rother et al., 2004; Li et al., 2004) inside and outside the object of interest, (d) region of interest (Kass et al., 1988; Mortensen and Barrett, 1995), (e) bounding boxes (Rother et al., 2004; Xu et al., 2017b; Agustsson et al., 2019) enclosing the object, and (e) clicks (Xu et al., 2016; Liew et al., 2017). User interactions marking background pixels are shown in red; green markings are used to indicate object enclosures or user interactions marking foreground pixels.

salient contour. As the contours evolve by minimizing their energy functional, they often *wiggle* and *slither,* which accounts for their name. Snakes continued to remain popular well after its debut. Variants of the original implementation such as finite element snakes (Cohen and Cohen, 1993), B-snakes (Menet et al., 1990; Blake and Isard, 1998), Fourier snakes (Staib and Duncan, 1992), T-Snakes (McInerney and Terzopoulos, 2000), and Diffusion snakes (Cremers et al., 2002) were proposed to decrease the sensitivity towards contour initialization and to increase robustness against noise. However, Snakes required a high degree of domain dependence and extensive user interaction to define an initial boundary. Segmentation results remained too dependent on the initialization. If the result was not satisfactory, the process must be repeated with a different initialization, or the boundary needed to be manually edited. Often this was tedious and time-consuming. Other avenues for correction included parameter adjustment, which is difficult for a naive user.

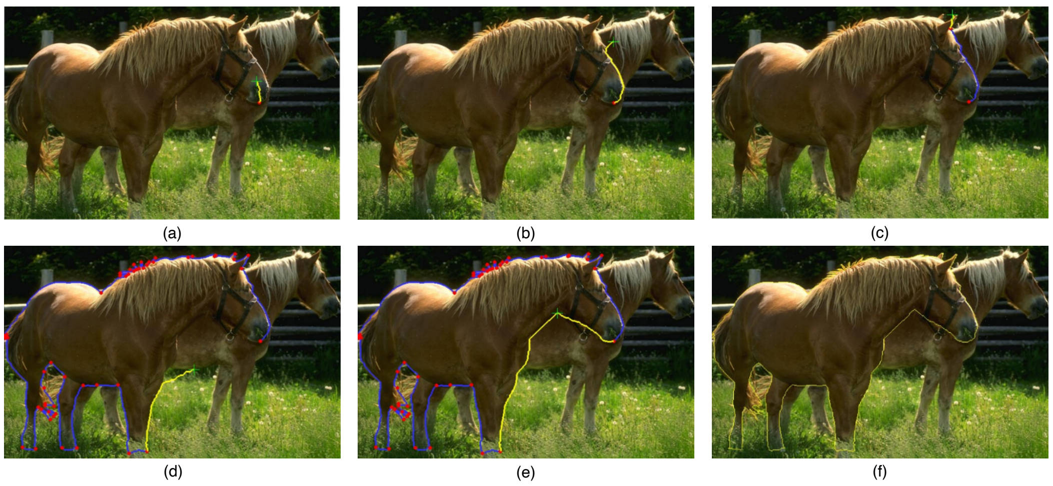


Figure 2.2: Intelligent Scissors. (a) At the onset, the user places a click on the desired object boundary (marked in red). (b-c) The user then proceeds to provide the next boundary click; the green cross-hair shows the live marker location, the yellow line shows the live contour segment. (c) Following the second click, a "set" or fixed segment boundary is generated (marked in blue) (d-e) The user continues tracing the object boundary placing additional boundary clicks till (f) the final object boundary is obtained. Image from Mortensen and Barrett (1998).

Intelligent Scissors. After Snakes (Kass et al., 1988), Intelligent Scissors (Mortensen and Barrett, 1995; Barrett and Mortensen, 1996; Stalling and Hege, 1996), also referred to as Live Wire (Barrett and Mortensen, 1997; Falcao et al., 1998), became popular as an interactive boundary tracing tool. Intelligent Scissors allows a user to choose a minimum contour by roughly tracing the object's boundary using an interactive terminal, e.g. the computer mouse. As the user traces the cursor along the boundary, the minimum cost path from the current cursor position to the last "seed" point is shown. The entire 2-D object boundary is specified via a set of live-wire segments in this manner. Intelligent Scissors share several similarities with Snakes. Both Snakes and Intelligent Scissors require user interaction and use similar boundary features and cost functions to find optimal boundaries. However, in literature, they are considered as competing methods as their methodologies differ from one another in several ways. First, Snakes iteratively compute a final optimal boundary by refining a single initial boundary approximation provided the user. In Intelligent Scissors, the user interactively select an optimal boundary segment from potentially all possible minimum cost paths; the complete boundary is obtained through incremental movements. Second, Snakes are globally optimal over the entire contour, whereas Intelligent Scissors boundaries are piecewise optimal (i.e., between seed points). Variants of both Snakes and

Intelligent Scissors achieved commercial success as part of popular software such as GIMP, Magnetic Lasso in Adobe Photoshop (Li et al., 2004).

Summary. While easier and more economical than just manually tracing individual pixels on the boundary, both Snakes (Kass et al., 1988) and Intelligent Scissors (Mortensen and Barrett, 1995) still demand a large amount of user attention. In images with highly textured regions, there exist many alternative minimal paths. Intelligent Scissors, in such instances, requires many user interactions. If a mistake is made, the user has to retrace the curve; for Snakes the user has to re-draw the entire contour again. Finally, once the initial boundary is specified, both Snakes and Intelligent Scissors are no longer helpful. Any errors at the end must be attended to at the pixel level using traditional selection tools.

2.2 Classical region-based approaches

Boundary-based methods paved the way for region-based methods. Region-based methods, such as interactive Graph cuts (Boykov and Jolly, 2001) and GrabCut (Rother et al., 2004), became popular as they typically demanded less precise hints from the user. Instead of enclosing regions (Kass et al., 1988) or pixel-accurate clicks on the boundary (Mortensen and Barrett, 1995), these approaches required hints in the form of few pixel clicks or scribbles (Rother et al., 2004). The popularity of region-based approaches was driven by the success of graph cuts (Ford Jr and Fulkerson, 1962; Goldberg and Tarjan, 1988; Boykov and Jolly, 2001) for a variety of computer vision tasks. Graph cut algorithms had already demonstrated great potential for solving many vision and graphics problems, such as image restoration, multi-view reconstruction, texture synthesis, etc. An additional benefit of graph cut algorithms (Boykov and Jolly, 2001) is that it seamlessly allows to incorporate both regional and boundary properties of the segment. Thus, graph cuts became the algorithm of choice for most newly proposed interactive segmentation methods. In these methods (Boykov and Jolly, 2001; Rother et al., 2004), the foreground and background seeds (user-provided hints) are treated as source and sink nodes for a max-flow min-cut operation in the Graph Cuts method. Using a max-flow min-cut algorithm, a set of edges with the minimum total weight is found and then returned as the object boundary.

Interactive graph cuts. In interactive graph cuts (Boykov and Jolly, 2001), the user imposes hard constraints for the final segmentation by indicating certain pixels (seeds) that have to be part of the object and certain pixels that have to be part of the background. These hard constraints provide clues on what the user intends to segment. The globally optimal instance boundary is then obtained using a fast min-cut max-flow algorithm (Boykov and Kolmogorov, 2004). The set of edges with

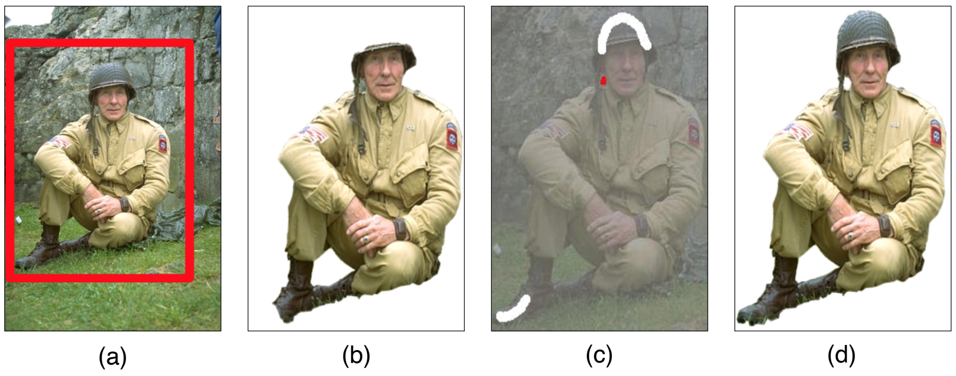
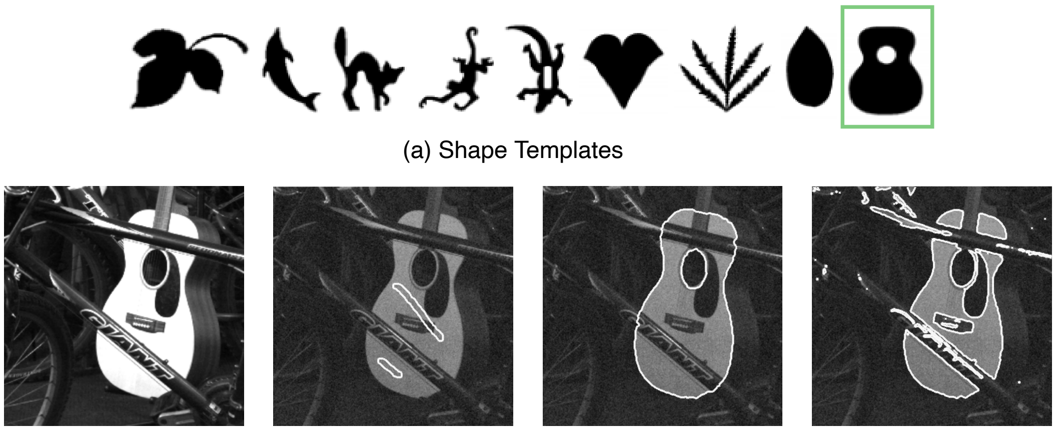


Figure 2.3: GrabCut. (a) The user marks the foreground with a bounding box. (b) Following the initial segmentation (c), user provides additional edits by marking the foreground and background with white and red brush respectively. (d) The process is repeated till the desired result is obtained. Image from Rother et al. (2004).

the minimum total weight is found and returned as the object boundary. Due to the algorithm's stability and strong mathematical foundation, interactive graph cuts became popular and led to several variants and extensions, e.g. GrabCut (Rother et al., 2004), Lazy Snapping (Li et al., 2004) etc. It should be noted that hard constraints were also an integral part of previous boundary-based interactive segmentation techniques. For example, in Intelligent Scissors (Mortensen and Barrett, 1995), the user had to indicate certain seed pixels where the segmentation boundary should pass. In Snakes (Kass et al., 1988), the users confined the object with an initial boundary approximation; pixels on the outside had to segmented as background.

GrabCut. GrabCut (Rother et al., 2004) is a pioneering example of iterative interactive segmentation; it extends interactive graph cuts (Boykov and Jolly, 2001) by introducing an iterative refinement scheme that uses graph cut for the intermediate updates. It starts with an user drawing a bounding box around the object of interest which serves as an initial approximation of the object labeling. At each iteration, it gathers and updates the foreground background color statistics (using Gaussian mixture models) according to the current prediction, and applies graph cut on the re-weighted graph to compute a newly refined mask (Fig. 2.3).

A serious difficulty across graph cut based methods is its bias towards producing segments with shorter boundaries; a *small cut* (Grady et al., 2005). This is also known as the *length* or *shrinking bias* (Price et al., 2010). The boundary term in graph-cut methods consists of a summation over the segmented regions' boundary. If the regional properties are weighted not as heavily as the boundary properties, the max-flow min-cut algorithm (Boykov and Kolmogorov, 2004) returns the smallest cut separating the seeds. It may return very small segmentation if a smaller number



(a) Shape Templates

(b) Guitar occluded by bicycle

(c) Initial user scribbles

(d) Segmentation with shape prior

(e) Segmentation *w/o* shape prior

Figure 2.4: Graph Cuts with shape prior. (a) Shape templates (b) The user intends to segment the guitar occluded by the bicycle. (c) The user provides initial scribbles to mark the target of interest. (d) Segmentation result obtained using the template in (a) as shape prior (d) Segmentation result obtained without using a shape prior. Image from Vu and Manjunath (2008).

of seeds are used or if the seeds are not well spread out across the object. This also causes problems by returning solutions where boundary takes a *shortcut* over a protruded section of the object in an attempt to minimize boundary length.

Graph Cuts *with* shape priors. Several methods (Freedman and Zhang, 2005; Slabaugh and Unal, 2005; Malcolm et al., 2007; Vu and Manjunath, 2008; Veksler, 2008; Das et al., 2006; Gulshan et al., 2010; Isack et al., 2018) incorporated shape priors into graph cut techniques to counteract the bias towards shorter boundaries. Such shape priors include elliptical (Slabaugh and Unal, 2005), convex blobs (Das et al., 2006), and star shape (Veksler, 2008; Gulshan et al., 2010; Isack et al., 2018) priors. Other GrabCut extensions include incorporating tightness priors for bounding boxes (Lempitsky et al., 2009). The work of Slabaugh and Unal (2005) was the first one to consider the use of shape priors in the form of ellipses in a graph cuts-based algorithms. Ellipses offer a simple yet descriptive shape that can model a wide range of objects, including anatomical structures like blood vessels and lymph nodes, the segmentation of which was the primary application in (Slabaugh and Unal, 2005). The work of (Das et al., 2006) propose a graph cut-based method for segmenting *compact* objects, i.e. objects with low perimeter to area ratio (Veksler, 2002). They modify the boundary term to incorporate bias towards larger objects. However, both these works, did not generalize to highly variable shapes. In a more generic approach towards shape prior for graph cut segmentation, (Veksler, 2008)

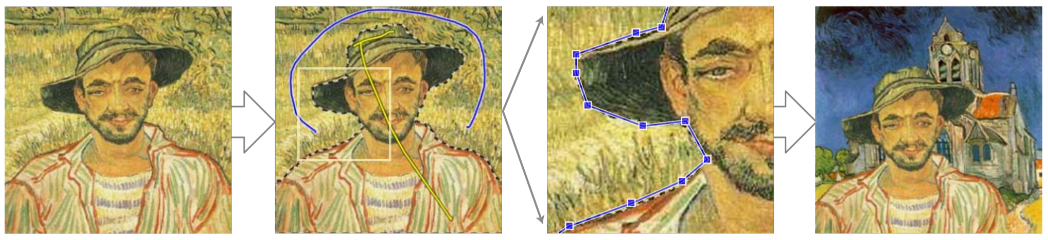


Figure 2.5: Lazy Snapping for digital image composition. (a) Input image (b) The user marks the object and background with yellow and blue scribbles respectively. (c) Based on the initial result, the user can additionally zoom in and adjust the current boundary by interacting with the polygon vertices in the zoomed-in view. (d) Once a satisfactory result is obtained, the object mask can be used for downstream tasks such as digital image composition. Image from Li et al. (2004).

incorporated the star shape prior. A major benefit of incorporating the shape constraints is that the objects were segmented more robustly and reliably. However, these techniques all require additional parameters or computation.

Lazy Snapping. For low-resolution images, GrabCut operated at an interactive speed (Li et al., 2004). However, its use for segmenting higher resolution images was limited by intense memory requirements and time complexity. To overcome this, the work of Lazy Snapping (Li et al., 2004) proposes a two-step coarse-to-fine approach for interactive instance segmentation. To mitigate the computational burden, the image is initially segmented using the watershed algorithm (Vincent and Soille, 1991). Different from interactive graph cuts (Boykov and Jolly, 2001) and GrabCut (Rother et al., 2004), the nodes in the graph cut optimization are now the segmented regions instead of the pixels. This approximation produces significantly improves the speed. The number of nodes and edges for the graph cut algorithm is reduced by more than 10 times (Li et al., 2004) compared to GrabCut. The final boundary edits are performed using a simple polygon framework, in similar spirit to Intelligent Scissors (Mortensen and Barrett, 1995). Lazy Snapping offers the advantages of both region-based (e.g. relaxed user seed specification) and boundary-based (e.g. pixel-level boundary editing) methods in an unified framework.

The work of Lombaert et al. (2005) also studied the application of graph cuts for high-resolution data. To reduce the computational burden, the approach performs graph cuts on a lower resolution. Next, it propagates the result to the higher resolution image by only computing the graph cuts in a narrow band surrounding the projected foreground background. As the graph cut runs only on the sub-graph in the the narrow band, the additional computation required at the higher resolution is significantly less than running it on the full graph. This use of a smaller graphs in all resolutions reduces the running time and memory consumption compared with the original graph cut algorithm.

Random Walks. In another graph-based approach (Grady et al., 2005), the author used random walks to determine the foreground-background pixel labels via label propagation on a weighted graph. Since this algorithm is not seeking the smallest boundary, it does not suffer from the *small cut* problem. After obtaining foreground and background (seed) pixels from users, the algorithm places a random walker at each unlabeled pixel. Each pixel is then assigned the label of the user-annotated pixel they first arrive at. The work of Sinop and Grady (2007) extended this formulation by proposing a framework that unifies random walk, graph cuts, and shortest path algorithms for the task of interactive segmentation. Compared to existing graph cut methods, random walk algorithms achieve better segmentation performance. However, the random walks algorithm lacks a global color distribution model, making it very sensitive to the location and the number of foreground and background seeds. Other extensions include the work of Price et al. (2010) which uses the geodesic distance as a unary for the graph cut optimization.

Summary. The primary drawback of these *classical* approaches is insufficient or rather lack of semantic or *objectness* priors. These methods rely on low-level features, such as color distributions and contrast, to model the object and background distributions. Therefore, their performance is restricted by the suitability of low-level features to distinguish between the foreground and background. As a result, for images with similar foreground and background appearances, complex textures, and difficult lighting conditions, the annotation effort increases significantly as these algorithms require users to label a large number of pixels (Xu et al., 2016).

2.3 FCN-based deep learning approaches

As with most of the other research in computer vision, image segmentation was greatly impacted by the arrival of deep neural network architectures. Deep neural networks (DNNs) learn a strong representation of objectness and have shown superior performance in distinguishing different objects across images (Russakovsky et al., 2015). The features learned by DNNs also proved to be highly transferable to other tasks such as semantic (Long et al., 2015) and instance segmentation (He et al., 2017). Deep semantic and instance segmentation approaches outperformed existing state-of-the-art algorithms with ease (Long et al., 2015; He et al., 2017). Not surprisingly, several researchers since then have focused on applying deep segmentation models to improve existing interactive segmentation frameworks.

The success of instance segmentation and semantic segmentation models, however, did not directly translate to interactive segmentation methods. *First*, semantic segmentation methods, such as Fully Convolutional Networks (FCNs) (Long et al., 2015), operates at class-level masks and not instance-level masks. *Second*, both semantic and instance segmentation approaches do not generalize to instances from unseen classes. This lack of generalization ability to unseen classes makes it impractical as one needs to train a model for each and every possible object class. Most importantly, existing fully automated instance and semantic segmentation methods do not respond to user inputs. While user-provided clicks can select an instance mask from generated segmentation, it was unclear how the algorithms would respond to additional user clicks.

FCNs inspired the first deep learning-based interactive instance segmentation framework. In the seminal work of Xu et al. (2016), the authors proposed an FCN-based architecture that accepts user-provided clicks as additional input along with the image. In this framework, to select an object instance in the image, users provide a "positive" click on the object of interest. Based on the initial prediction, users can then iteratively provide "positive" and "negative" clicks to add object pixels misclassified as background and remove background pixels misclassified as the object. The process is repeated until a satisfactory result is obtained.

The usability of user-provided positive and negative clicks is restricted because of their sparsity. In order to efficiently leverage such user-provided clicks, in (Xu et al., 2016), they are converted to *guidance maps* using Euclidean distance transformation. The 2-channel *guidance maps*, one channel each for the "positive" and "negative" clicks, are then concatenated with the 3-channel RGB image, and fed as input to a FCN (Long et al., 2015). Typically, the weights of such FCNs are initialized with that of the state-of-the-art network on semantic segmentation FCN-8s (Long et al., 2015). The network is trained with only two labels - *object* and *background*. Since obtaining clicks from actual users for training the network require significant effort, user clicks are sampled using several heuristics to generate image - user interactions training tuples. The segmentation model is fine-tuned on many of these pairs. At test time, clicks are sampled and added one by one based on the errors of the currently predicted mask. The process is repeated until a satisfactory segmentation mask is obtained; for challenging datasets such as Pascal VOC 2012 (Everingham et al., 2010) and MS COCO Lin et al. (2014), the generated masks should have a minimum of 85% mIoU *w.r.t* the ground truth. In Xu et al. (2016), users can interact with the segmentation models only via point clicks. Additionally, user clicks added during refinement often fail to improve the prediction significantly. A following work (Xu et al., 2017b) investigated the use of loose bounding boxes and other arbitrary shaped closed curves (e.g. ellipses,

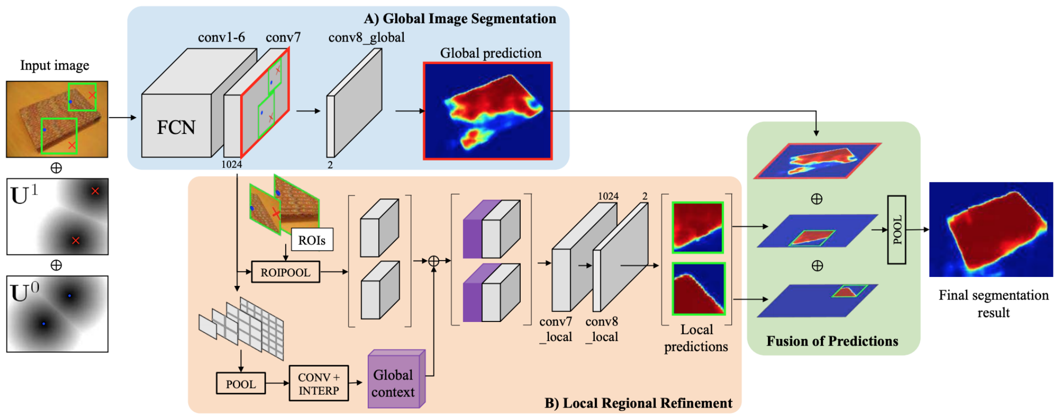


Figure 2.6: Framework of RIS-Net (Liew et al., 2017). In order for refinement clicks to significantly improve the generated segmentation mask, RIS-Net proposed a two-branch network consisting of a global branch for the entire image segmentation and a local branch for local regional refinement. The global branch is similar to the framework of Xu et al. (2016); the input image with the guidance maps are fed forward through a FCN to obtain a coarse segmentation. In the local branch, feature descriptors for the click pair are extracted via the ROI pooling layer which are then enriched with global feature descriptors. Finally, both global and local predictions are fused to output the final segmentation mask. Image from Liew et al. (2017).

circles) for interactive segmentation. Such arbitrary shapes are easier to mark and reduces annotation effort. Given an arbitrary shape loosely outlining the object, the algorithm first transforms it into an Euclidean distance map. Similar to (Xu et al., 2016), a FCN then predicts the foreground background from the concatenated image-distance map input. However, in (Xu et al., 2017b), users cannot provide additional corrective clicks to improve upon the initial segmentation.

In order to fully leverage the information in refinement clicks, the work of Liew et al. (2017) proposes a two-branch network - it includes a global branch producing coarse global predictions and a local branch utilizing multi-scale spatial pyramid features to make refined local predictions; the final prediction is the combined results from the two branches. Both the frameworks of (Xu et al., 2016) and (Liew et al., 2017) apply graph cut optimization to the output segmentation mask from the FCN model.

While the frameworks in (Xu et al., 2016, 2017b; Liew et al., 2017) varied in network design and mode of user interaction (clicks *vs.* arbitrarily-shaped enclosures), all methods used Euclidean distance transform to encode the user-provided clicks. The work of Benard and Gygli (2017) empirically showed that encoding user clicks using Gaussians with a small standard deviation over Euclidean distance maps (Xu et al., 2016; Liew et al., 2017) led to performance improvement. A more recent work (Benenson et al., 2019) observed that encoding user clicks as small binary disks is more effective than the Gaussian transformation.

Across these previous methods (Liew et al., 2017; Xu et al., 2016; Benard and Gygli, 2017), the click sampling strategies adopted during training and testing are different. User clicks during training are sampled independent of the training error following heuristics outlined in (Xu et al., 2016). However, at test time, these methods base their click sampling strategy to mimic the clicking pattern of a human annotator; clicks are sampled iteratively based on the errors of the predicted mask. Predetermined click placements during training with no regards to the network prediction errors has an adversely affects the model accuracy. To address this discrepancy in click sampling during train and test time, the work of Mahadevan et al. (2018) proposed an iterative training algorithm where it samples training user clicks progressively based on the misclassification in the network prediction following each epoch. This led to a better segmentation performance without requiring post-processing via graph cuts (Xu et al., 2016; Liew et al., 2017; Benard and Gygli, 2017). This training algorithm was adopted across most of the interactive segmentation approaches that followed (Liew et al., 2019; Lin et al., 2020).

The work of Maninis et al. (2018) introduced an interactive approach for foreground segmentation from the extreme points (i.e., left-, right-, top- and bottom-most pixels) of the object. Here, users mark the extreme points on the object boundaries; these extreme points are then encoded using Gaussians and then concatenated with the 3-channel RGB channel. The 4-channel input is passed through a segmentation network to produce the final prediction. While the approach typically produces high quality segments, it is difficult to refine the unsatisfactory segments with additional clicks (Mahadevan et al., 2018). Similar in spirit to Maninis et al. (2018), the work of Le et al. (2018) proposed an interactive approach where the users click on the boundary of the target object. However, the boundary clicks in (Le et al., 2018) is not limited to the extreme points.

Fig. 2.7 shows the two-stream framework for interactive segmentation. Different to prior approaches, the two-stream network uses two dedicated streams to extract features from the image and the user-interactions. The extracted features are fused at different layers of the network enabling both early and late fusion of the image and user interactions. This form of late fusion allows the user-provided cues to have more impact on the predicted mask as the abstraction, in the form of stacked network layers, between the prediction and the user interaction is reduced. It

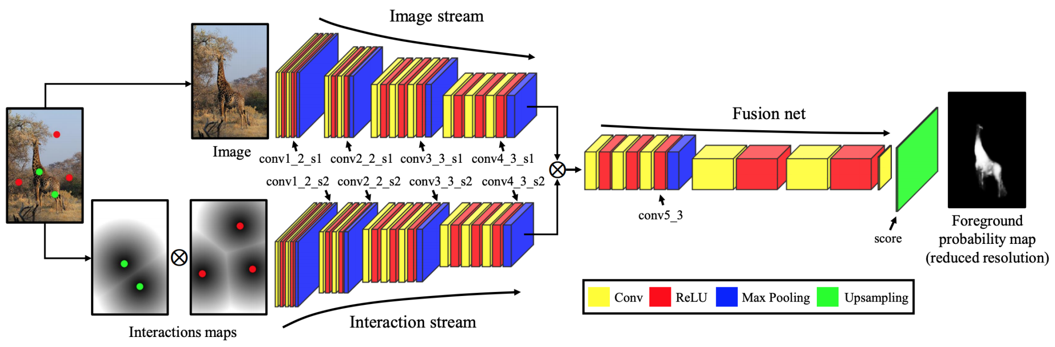


Figure 2.7: The framework of Hu et al. (2019) utilizes two separate networks but with identical architecture to extract features from the image and the guidance maps. The network consists of the first 10 convolutional layers of the VGG-16 (Simonyan and Zisserman, 2014) network. Features from both the image and the interaction streams are concatenated; a fusion network then is used to predict the segmentation mask from the concatenated feature maps. Image from Hu et al. (2019).

should be noted that the work of Liew et al. (2017) also uses a two-stream network; however, it only performs an early fusion of the image and the guidance maps generated from user interactions.

In interactive instance segmentation, there can be situations where the algorithm fails to assess the diversity in user expectations particularly when the target object is a part of another object. Consider a scenario when an user intends to segment the tie worn by a person. The algorithm can incorrectly generate a mask for the person thereby requiring excessive user input to correct the segmentation result. To address such ambiguities, the works of (Li et al., 2018; Liew et al., 2019) proposed interactive frameworks which generates multiple hypothesis segmentations and selects a more accurate choice among them. The work of Li et al. (2018) demonstrated that training the network to generate diverse segmentation given the user click improves the segmentation performance. However, the unconstrained training framework of Li et al. (2018) meant that generated diverse segmentations were either not too different from one another or not meaningful. The approach in Liew et al. (2019) improved upon this by imposing a set of scale priors to produce candidate prediction masks of different scales given the user click. An objectness classifier is used on top of the diverse segmentations to generate meaningful object masks for the user to select from (Fig. 2.8).

Other notable FCN-based interactive segmentation frameworks include (Jang and Kim, 2019; Sofiiuk et al., 2020). The work of Jang and Kim (2019) observed

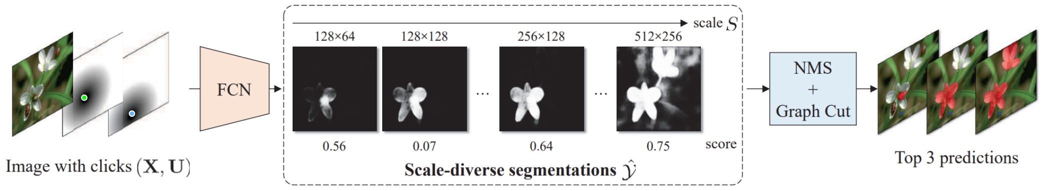


Figure 2.8: Consider the scenario where the user places a click on the bug on the flower. It is unclear from the click itself whether the user desires the segmentation mask for the bug or for the flower. To circumvent this, the segmentation network in Liew et al. (2019) outputs a set of candidate segmentation masks of various scales and their corresponding objectness scores; NMS further prunes the set of candidates and presents the top K predictions to the user. Based on the user intention, the user can either opt for the mask corresponding to the bug, or the flower with an additional click. Image from Liew et al. (2019).

that across prior works, it is not guaranteed that the user-provided pixel annotation matches the network output,i.e. the user selects an object with a positive click, however in the resulting segmentation mask, that click is assigned a background label and vice-versa. In order to ensure consistent segmentation output, during inference, the framework of Jang and Kim (2019) performs backpropagation on the network activations iteratively until all user-clicked pixels have correct labels in the output segmentation mask; the network weights are kept unchanged. The work of Sofiiuk et al. (2020) expand upon this framework by reducing the computational cost resulting from the optimization at inference time.

In all the mentioned approaches, user provided clicks are treated indiscriminately by the FCN. The work of Lin et al. (2020) pointed out the unique nature of the first click; it not only indicates the location of the target instance but also provides global context for the target object. Ensuing clicks serve to improve the segmentation obtained after the first click. Accordingly, in their approach, a separate module is proposed for utilizing the guidance from the first interaction click.

Summary. FCN-based approaches paved the way for successful interactive instance segmentation methods. The availability of state-of-the-art semantic segmentation models and the availability of annotated instances in images continues to be a significant force in driving down user-annotation efforts. Simply by adapting state-of-the-art models, it was possible to improve the segmentation performance significantly. Furthermore, different from early classical approaches, which relied on bounding boxes (Rother et al., 2004) and scribbles (Kass et al., 1988; Mortensen and Barrett, 1995; Li et al., 2004), the FCN-based frameworks use pixel clicks which are cheaper to annotate. Empirically, these methods demonstrated that a single click is often sufficient to obtain a good segmentation mask for specific instances.

One common drawback, however, applicable for a majority of the approaches is that the training and evaluation of these methods rely on synthetically generated clicks following the heuristics established in (Xu et al., 2016). User clicks are understandably costly to annotate. Especially with datasets like Pascal VOC 2012 and MS COCO, where the number of instances can vary from the order of tens of thousands to hundreds of thousands, the cost of acquiring actual user click annotation can be prohibitive.

Finally, majority of FCN-based methods demonstrate their efficacy based on the evaluation framework proposed in (Xu et al., 2016). While it ensures fairness in comparing methods, it often precludes comparison in terms of actual user time, e.g. in seconds, it takes for the algorithm to reach the desired quality in producing segmentation masks. A time-based evaluation framework is critical as on-boarding state-of-the-art segmentation models can improve performance significantly but at the cost of extra computation. Time-based comparisons can currently be observed across current interactive video object segmentation frameworks where algorithms are judged on the quality of the generated segmentation in a specified time window, e.g. mIoU@60 seconds (Oh et al., 2019; Heo et al., 2020, 2021).

2.4 Mask R-CNN -based deep learning approaches

A majority of the FCN-based approaches interactively segment only one object at a time. For full image segmentation (Andriluka et al., 2018; Agustsson et al., 2019), i.e. interactively segmenting all objects in the scene, such FCN-based approaches would require multiple passes. This stems from the fact that the FCN backbones used in such frameworks are trained for the task of instance-agnostic semantic segmentation. Different to FCNs, Mask R-CNN (He et al., 2017) are trained for the task of automated instance segmentation.

In (Andriluka et al., 2018), Mask R-CNN (He et al., 2017) is used to create a fixed pool of proposal segments; the human annotator then performs full image segmentation by assigning proposal segments to the individual objects. The work of Agustsson et al. (2019) starts with annotating the four extreme points for individual objects (Maninis et al., 2018). Instead of clicks (Xu et al., 2016; Mahadevan et al., 2018), errors in network predictions are iteratively refined with corrective scribbles. The approach of Andriluka et al. (2020) extends the segmentation to non-object categories (e.g., sky, grass) as well.

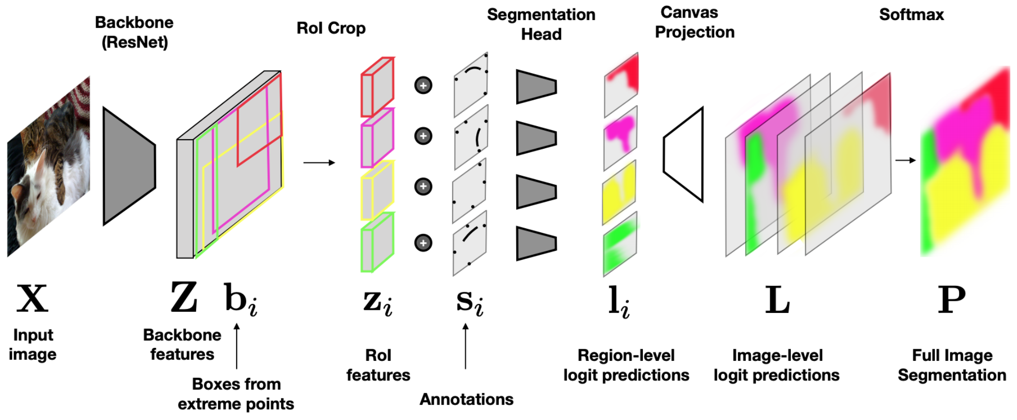


Figure 2.9: The framework of Agustsson et al. (2019) starts with an image input to the Mask R-CNN model (He et al., 2017). Next, RoI features for the objects are obtained directly via cropping by leveraging the user-provided extreme points. Additional corrective scribbles are concatenated with the RoI features and then processed by the segmentation head. Individual region predictions are projected into the image canvas.Image from Agustsson et al. (2019).

Summary. Mask R-CNN -based approaches offers certain advantages over the FCN-based interactive frameworks. Conventionally, FCN-based approaches focus on segmenting one instance in the image at a time. Segmenting each instance, for example, in Fig. 2.9, a conventional FCN-based approach would require four separate passes, with each pass requiring additional refinement clicks from the user. In addition to requiring multiple passes, in current FCN-based approaches, the interactions meant for a single instance are not leveraged for the other instances in the scene. Mask R-CNN-based approaches overcome these two limiting factor of FCN-based approaches. In Mask R-CNN -based approaches, corrective refinements for a specific instance can be leveraged as a cue for the remaining instances in the image (Agustsson et al., 2019).

With the exception of (Andriluka et al., 2020), a major drawback of existing Mask R-CNN -based approaches (Andriluka et al., 2018; Agustsson et al., 2019) is their inability to generalize to novel object classes. Although withheld from (simulated) users during the interactive, class labels are nonetheless used to obtain class-specific predictions during training and inference. Unlike FCN-based approaches which performs class-agnostic binary segmentation, the class labels are indeed an integral part of the network design for the Mask R-CNN -based approaches.

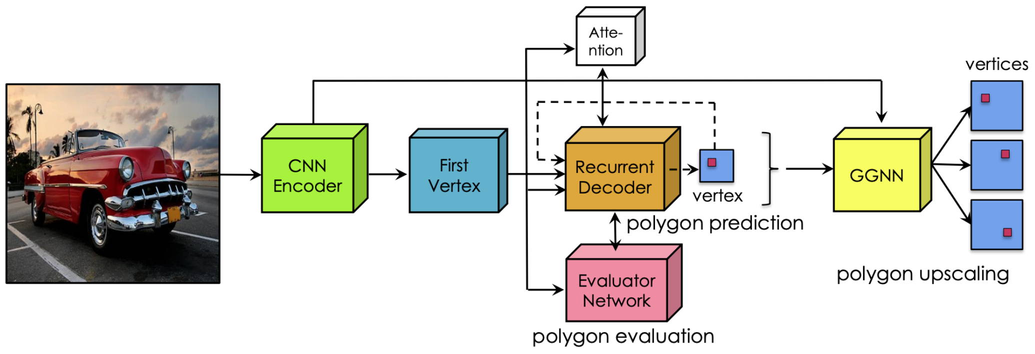


Figure 2.10: In Acuna et al. (2018), the image is first processed by the CNN to predict the first vertex of the polygon. This initial vertex along with the generated image features are fed to a RNN which outputs polygon vertices at each time step. An evaluator network selects the best polygon from a set of candidates proposed by the RNN decoder. Finally, a graph neural network aligns the polygon vertices and generates the image-sized output. Image from Acuna et al. (2018).

2.5 RNN-based deep learning approaches

Another group of approaches formulate the task of interactive instance segmentation as one of polygon prediction problem (Castrejon et al., 2017; Acuna et al., 2018; Ling et al., 2019). In (Castrejon et al., 2017; Acuna et al., 2018), a recurrent neural network (RNN) is used to predict the vertices of a polygon outlining the target instance sequentially. The annotator can intervene whenever an error occurs, by correcting the misplaced vertex. The network continues its prediction by conditioning on the correction. The work of Ling et al. (2019) parameterize objects using polygon or splines and uses graph convolution networks (GCNs) for performing the segmentation.

Summary. Different from FCN-based and Mask R-CNN-based approaches, RNN-based approaches do not produce dense per-pixel segmentation masks as outputs. Instead, these methods predict a polygon vertices or spline to enclose an object; such sparser outputs are desirable given the recurrent nature of the models. However, similar to Mask R-CNN-based approaches, RNN-based approaches are limited to the object classes that the models observed during training and cannot generalize to novel object classes during test time.

Chapter 3

Background

In this chapter, we discuss the key components that constitute our proposed interactive segmentation frameworks. We begin with briefly discussing Convolutional Neural Networks (CNNs) and the functional units that constitute them. CNN's are ubiquitous across deep learning-based interactive frameworks; they act as feature extraction backbones for semantic segmentation and instance segmentation approaches. Next, we discuss Fully Convolutional Networks or FCNs; our proposed frameworks rely on such pre-trained FCN models to yield per-pixel outputs. We then provide a brief overview of the superpixel and object proposal generation algorithms that constitute the pre-processing step for content-aware user click encodings. Finally, we conclude the chapter with an overview of the image datasets and the evaluation metrics used for benchmarking interactive image segmentation approaches.

3.1 Convolutional Neural Networks (CNNs)

CNNs have enjoyed great success across various large-scale image classification challenges (Russakovsky et al., 2015). Their name stems from the most significant operation in the network - convolution. CNNs can be viewed as a sequence of layers; these layers can be grouped into two primary categories - (i) feature extraction layers and (ii) classification layers. In the feature extraction layers, CNNs perform a series of convolution and non-linear operations (e.g., tanh/ReLU activations and spatial pooling operations) over different resolutions of the feature maps. Feature extraction layers in succession enable CNNs to extract and integrate low (e.g., edges), mid, and high-level (e.g., objectness) features (Zeiler et al., 2011). The depth, i.e., the number of stacked convolutional layers in the network, decides the levels/hierarchies of features encoded by the CNNs; the deeper, the better (He et al., 2016). The classification layers primarily consist of fully connected layers, which learn non-linear combinations of the extracted features. CNNs are integral to the state-of-the-art performance of fully convolutional networks (FCNs) for image segmentation tasks such as semantic segmentation and interactive image segmentation. In this section, we begin the discussion with what constitutes a CNN. Next, we look into two CNN architectures - VGG-16 (Simonyan and Zisserman, 2014), and ResNet-101 He et al. (2016). VGG-16 and ResNet-101 constitute the backbone of our FCN-8s (Long et al., 2015) and Deeplabv2 (Chen et al., 2018b) semantic segmentation models which we adapt and use in our interactive frameworks (Chapters 4, 5, 6, and 7).

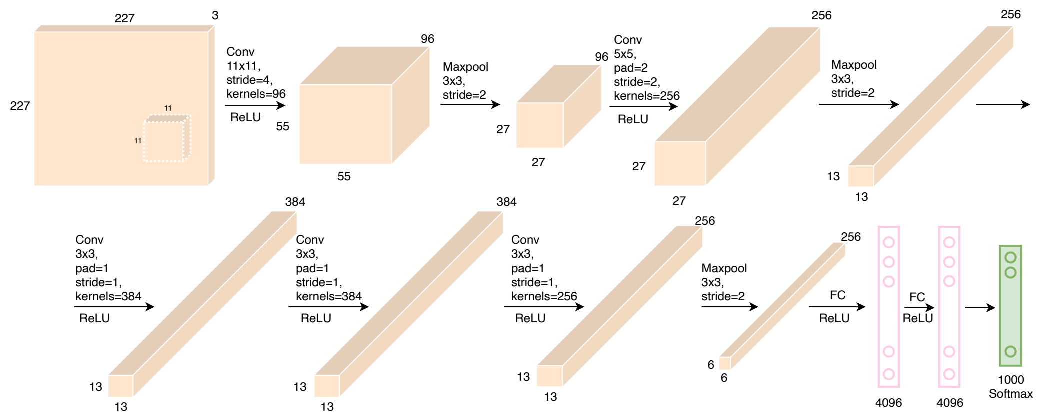


Figure 3.1: Example of a CNN architecture for image classification. Here, the CNN receives a RGB image as input and produces as output the probability of the image belonging to a particular class over 1000 classes (Russakovsky et al., 2015).

3.1.1 Components

Convolution. In CNNs, the convolution operation serves as a means to extract features from an 3-D input feature map (or an image for the first layer). The convolution of an input feature map $\mathbf{f}_{in} \in \mathbb{R}^{h \times w \times d_1}$ with d_2 convolution kernels of spatial extent $k_1 \times k_2$ results in a feature map $\mathbf{f}_{out}$ of dimension,

$$\left\lfloor \frac{h - k_1 + 2p}{s} + 1 \right\rfloor \times \left\lfloor \frac{w - k_2 + 2p}{s} + 1 \right\rfloor \times d_2 \tag{3.1}$$

As evident from Eqn. 3.1, the output dimension is controlled by five hyperparameters - number of filters d_2, spatial extent of the filter parameterised by k_1 and k_2, stride s, and padding p. For square kernels, we have $k_1{=}k_2$. Stride s defines the step size with which the convolution kernel moves each time. A stride of 1 means the kernel slides pixel-by-pixel; the larger the stride, the smaller the spatial output. Padding adds a layer of zero-pixel values surrounding the input feature map. Padding prevents the output feature map from shrinking.

Convolution offers several advantages, such as sparse interactions, parameter sharing, and equivariant representations (Goodfellow et al., 2016). Traditionally, in multilayer perceptrons or MLPs, each input unit interacts with each output unit via a unique set of network weights. However, in CNNs, the convolution kernel interacts with the input units *sparsely*; the output units typically interact with only a subset of the input units. Additionally, unlike MLPs, the same kernel weights are applied at every position of the input. Overall, it results in fewer parameters and keeps the model size in check. Finally, parameter sharing enables the convolution operation to be equivariant to translation. Conventionally, non-linear activations follow the convolution operation. ReLU (Maas et al., 2013) is such an example of a non-linear activation.

$$\mathrm{ReLU}(x) = \max(0, x) \tag{3.2}$$

Pooling. In CNNs, pooling operations typically take place in between CNN layers or *blocks* (see Fig. 3.2). The pooling operation involves sliding a filter individual channels of the input feature map and summarizing the features covered by the window. These operations are parameterised by the kernel size and the stride; a common example is a 2×2 window with a stride of 2. More formally, the output feature map dimension obtained after a pooling operation with kernel $k_1 \times k_2$ and stride s on an input feature map $\mathbf{f}_{in}$ of dimension $h \times w \times d_1$ is $\mathbf{f}_{out}$ of dimension,

$$\left\lfloor \frac{h - k_1}{s} + 1 \right\rfloor \times \left\lfloor \frac{w - k_2}{s} + 1 \right\rfloor \times d_1 \tag{3.3}$$

The commonly used pooling operations include average and max-pooling; average pooling replaces the feature map values with an average value computed over the pooling window. Likewise, the max-pooling operation replaces the feature map values with the maximum value within the neighborhood.

Max-pooling operation with non-overlapping strides performs several key functions. As evident from the stride, the pooling operations reduce the memory footprint of the intermediate feature maps. Next, it increases the receptive field of subsequent convolutions operations; 2×2 max-pooling with a stride of 2 increases the receptive field by a factor of 2. This allows the deeper layers of the network to discern high-level semantic information. Finally, for image classification tasks, "*whether an object exists?*" supersedes "*where does the object exist?*". With the max-pooling operation, the dominant feature from a region is always preserved making the network robust to small translations of the features in the input map.

Fully Connected (FC) layers. The fully-connected (FC) layers constitute the last few layers of CNNs. Different to convolution layers where feature responses are computed in local neighborhoods, every output response in an FC layer is computed by using the values of all the input feature map locations. Implementation-wise, FC layers are simply feed forward neural networks which learn non-linear combinations of extracted features from the convolutional layers. After the FC layers, the final layer in image classification CNNs use the softmax activation function to get probabilities of the input belonging to a particular category (classification).

Batch normalization. transforms every unit in a feature map to have zero mean and have unit variance. Batch normalization layers behave differently during training and inference. During training, the batch normalization (BN) layer normalizes the channel-wise output using the mean and the standard deviation of the current batch. During inference, the BN layer performs the channel-wise normalization using the moving average of the mean and standard deviation that the deep learning model observed during training. With batch normalization layers, it is possible to train deep neural networks using much higher learning rates while being less sensitive to parameter initialization (Ioffe and Szegedy, 2015). Conventionally, such layers are positioned before the non-linear activations (He et al., 2016). Generally, using batch normalization layers have been shown to improve the performance of deep learning models (Ioffe and Szegedy, 2015).

Training. CNN's are parameterized using trainable weights in the order of hundreds of millions (Simonyan and Zisserman, 2014). To find an optimal set of values for these weights, CNNs are trained to minimize a loss function. Depending on the task at hand, the loss function quantifies the discrepancy between the network output and the desired output specified in the ground truth, e.g., cross-entropy loss for image classification. Ideally, if the prediction deviates too much from the ground truth, the loss function should increase in value. Gradually, as the network updates its weights via some optimization, the value of the loss function decreases.

More formally, let the loss function be $\mathcal{L}(\mathbf{y}, \hat{\mathbf{y}})$ where $\mathbf{y}$ and $\hat{\mathbf{y}}$ denote the ground truth and the network prediction respectively. The optimal set of weights $\mathbf{w}$ are obtained by minimizing the empirical risk over the available training data $\mathcal{D}$ with N samples,

$$\mathcal{L}_{\mathcal{D}}(\mathbf{w}) = \frac{1}{N}\sum_{i=1}^{N}\mathcal{L}(\mathbf{y}_i, \hat{\mathbf{y}}_i) \tag{3.4}$$

The minimization of Eqn. 3.4 is performed via gradient descent on the total empirical loss. In practice, large-scale datasets are difficult to fit into memory. Hence gradient updates are performed after seeing only a few training samples in a mini-batch using Stochastic Gradient Descent (SGD). The weight update rule is given by,

$$\mathbf{w} \leftarrow \mathbf{w} - \alpha\nabla\mathcal{L}_d(\mathbf{w}) \tag{3.5}$$

where $d \subset \mathcal{D}$, α is the learning rate and $\nabla\mathcal{L}_d(\mathbf{w})$ denotes the weight gradients $w.r.t$ the loss $\mathcal{L}_d$ computed over the mini-batch $d = \{x_i, y_i\}_{i=1}^{K}; K << N$. We refer the reader to Goodfellow et al. (2016) for further details on CNN architectures and training.

3.1.2 VGG-16

AlexNet (Krizhevsky et al., 2012) demonstrated that state-of-the-art performance on challenging large-scale datasets such as ImageNet (Russakovsky et al., 2015) can be achieved using *deep* CNNs. However, such architectures typically use large

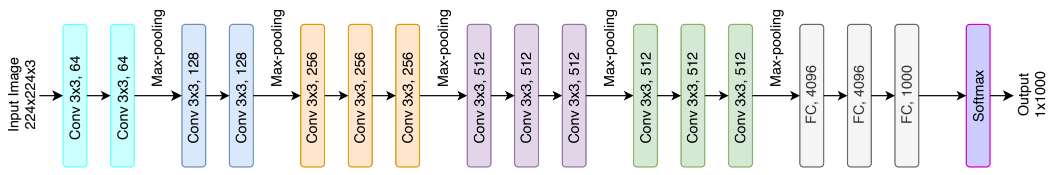


Figure 3.2: The VGG-16 (Simonyan and Zisserman, 2014) network consists of 5 convolution blocks with a total of 16 layers. At the end of each block, max-pooling operation downsamples the feature map along the spatial axes ($h{\times}w \rightarrow \frac{h}{2}{\times}\frac{w}{2}$) by a factor of 2. Here we show a image classification use-case; the network receives as input a RGB image which it classifies into one of the 1000 possible classes in the used dataset.

convolution kernels, such as 11×11 in AlexNet (Krizhevsky et al., 2012). This comes at the cost of increased computation. Unlike AlexNet, VGG (Simonyan and Zisserman, 2014) uses convolution filters with small kernels of size 3×3 with a stride of 1 for all convolutional layers in the network (Fig. 3.2).

Stacking multiple layers allows VGG to have the same receptive field as CNNs with large kernels but less depth, e.g., stacking two 3×3 convolutions, without spatial pooling operation, has an effective receptive field of 5×5; three such layers result in a 7×7 receptive field. However, having 3 layers of 3×3 over 7×7 offers two key advantages. First, instead of a single non-linear activation layer, the architecture can now incorporate 3 non-linear layers making the network more discriminative. Next, it requires fewer parameters. Next, a 7×7 kernel is parameterised by 49 network weights whereas 3 layers of 3×3 are parameterised by $3{\times}3^2 = 27$ parameters. This allows the VGG architecture to grow in depth without a parameter explosion.

3.1.3 ResNet-101

VGG-16 (Simonyan and Zisserman, 2014) demonstrated that increasing the number of stacked layers can improve the performance of deep CNNs which raises the question "*Is designing better CNN architectures as trivial as stacking more convolutional layers?*" Ideally, if the additional layers can perform identity mappings between the input and output feature maps, a CNN with more layers should not have training error greater than it's shallower counterpart. Unfortunately, indiscriminate stacking of convolutional layers to increase depth eventually leads to performance degradation which underlines the difficulties that these additional layers have in approximating identity mappings. The work of (He et al., 2016) address the degradation problem by introducing a deep residual learning framework. The building block of such a framework is shown in Fig. 3.3. Given an input feature map $\mathbf{f}_{in}$, each building block

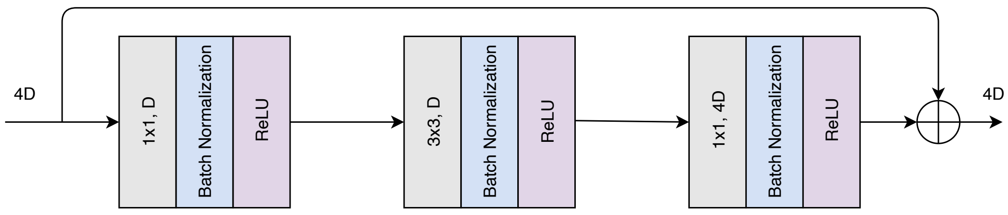


Figure 3.3: Bottleneck layer in ResNet-50/101/152 (He et al., 2016) utilizes 1×1 convolutions to create a bottleneck in the residual connection. This bottleneck layer reduces the number of parameters and lowers the computation cost and allows the network to increase in depth without significant increase in the number of trainable parameters. Here, D refers to the number of channels in the feature maps and $\oplus$ refers to the element-wise operation.

realizes a non-linear mapping $\mathcal{F}(\mathbf{f}_{in}) + \mathbf{f}_{in}$ of the input feature. With such a formulation, if $\mathbf{f}_{in}$ provides an optimal mapping, the weights of the *residual* layers are simply driven to zero by the optimization algorithm. Such a formulation can be realized by feed-forward neural networks with shortcut or skip connections. In the residual formulation, skip connections perform the identity mapping; such identity connections neither incur extra parameter nor computational complexity.

3.2 Fully Convolutional Networks (FCNs)

Fully Convolutional Networks or FCNs are a class of neural networks consisting only of convolution blocks (convolution, non-linearity, pooling) and no fully connected layers. Different to CNNs, FCNs can operate on inputs of any arbitrary dimension, and produce correspondingly-sized outputs. Popular FCN models include FCN-8s (Long et al., 2015) and Deeplabv2 (Chen et al., 2018b) for semantic segmentation. These semantic segmentation models are trained in an end-to-end fashion to minimize the *per-pixel* cross entropy loss.

3.2.1 FCN-8s

Deep CNNs trained on large scale image datasets yield powerful models for challenging computer vision tasks. However, conventionally the outputs of CNNs are limited to coarse prediction, e.g. class labels for images (Simonyan and Zisserman, 2014; He et al., 2016). Additionally, input image dimension are dictated by the kernel sizes in the fully connected layers; the kernel size should match the spatial dimension of the input feature maps. However, for several computer vision applications, e.g. semantic segmentation (Long et al., 2015), instance segmentation (He et al., 2017), and interactive instance segmentation (Xu et al., 2016), dense output in the form of per-pixel

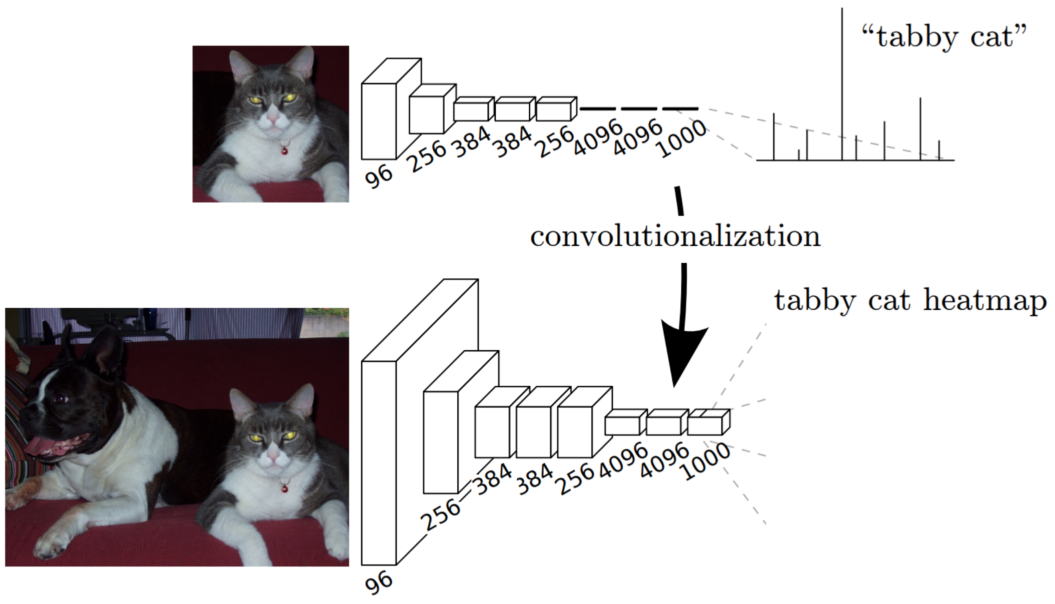


Figure 3.4: The framework of FCN (Long et al., 2015) transforms the fully connected layers of image classification CNNs into convolution layers. This enables FCNs to dense output in the form of per-pixel class prediction instead of an image-level prediction. Image from Long et al. (2015).

predictions are desired. Fully Convolutional Networks or FCNs (Long et al., 2015) are able to achieve this. The work of FCN-8s (Long et al., 2015) was the first to propose a FCN capable of being trained in a fully supervised end-to-end manner by adapting and extending existing image classification models, e.g. VGG (Simonyan and Zisserman, 2014). Fully connected (FC) layers in such deep CNN models accept fixed-size feature maps and disregard the spatial coordinates. Implementation-wise, such FC layers can be viewed as convolutions with kernels that span the spatial extent of the input. By doing away with the feature map *flattening* in the fully connected layers, the framework of Long et al. (2015) was able to predict class label at each individual pixel (after upsampling) (Fig. 3.4).

3.2.2 Deeplabv2

FCNs for semantic segmentation primarily use an encoder-decoder architecture. The encoders of choice are deep CNNs, e.g. VGG-16 in FCN-8s (Long et al., 2015). However, these deep CNNs originally devised for image classification aggressively downsample input images and ensuing feature maps to fit the dimensions of the inner fully connected layers. In FCN-8s, the VGG encoder reduces the spatial resolution by a factor of 32. Accordingly, FCN-8s has to rely on skip connections to preserve

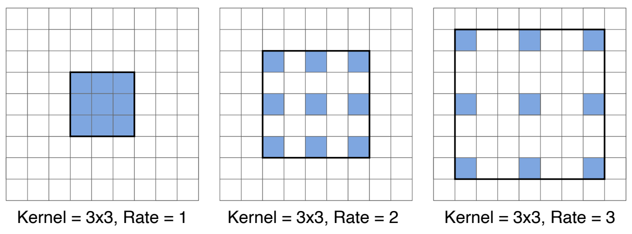


Figure 3.5: *Atrous* convolution (Holschneider et al., 1990) introduces an additional parameter "*rate*" to the convolution operation. This defines the spacing between the weights in the convolution kernels. *Atrous* convolution enables Deeplab (Liang-Chieh et al., 2015) to have a large field-of-view while keeping the number of parameters fixed. With a rate of 2, a 3×3 kernel has the same field-of-view as that of a 5×5 kernel, while only using 9 parameters instead of 25.

feature map resolution. In order to preserve feature resolution, Deeplab proposes the use of atrous convolutions (Liang-Chieh et al., 2015). Atrous (or dilated) convolutions are regular convolutions with a factor "*rate*" that allows us to expand the filter's field of view (Fig. 3.5). With atrous convolution, Deeplab is able to extract multi-scale features without feature map downsampling. Deeplabv2 (Chen et al., 2018b) introduces the idea of atrous spatial pyramid pooling (ASPP) into the Deeplab framework. ASPP adds multiple parallel atrous convolutions with different dilation rates; the resultant feature maps are processed in separate parallel branches and are then fused together to generate the final result (Fig. 3.6). ASPP allows the Deeplab framework to account for the varying object sizes.

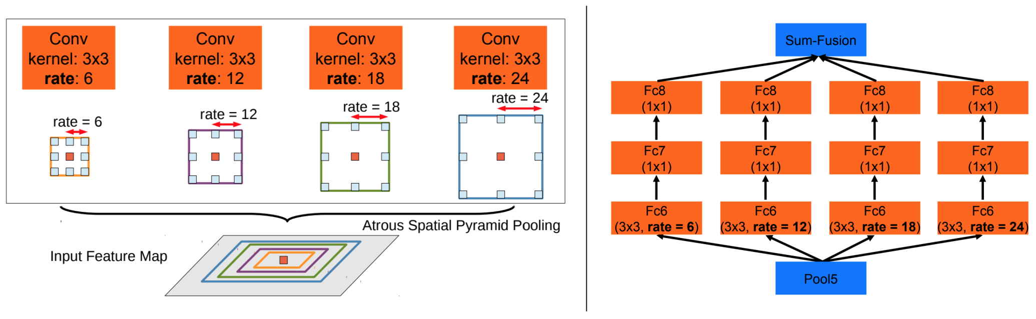


Figure 3.6: Deeplabv2 (Chen et al., 2018b) introduces the concept of ASPP into the Deeplab framework. Here, multiple parallel branches perform *atrous* convolution with different rates. This allows the incorporation of multi-scale information for a single pixel, as shown on the left. In the right, the implementation of ASPP framework within the VGG network is shown. Image from Chen et al. (2018b).

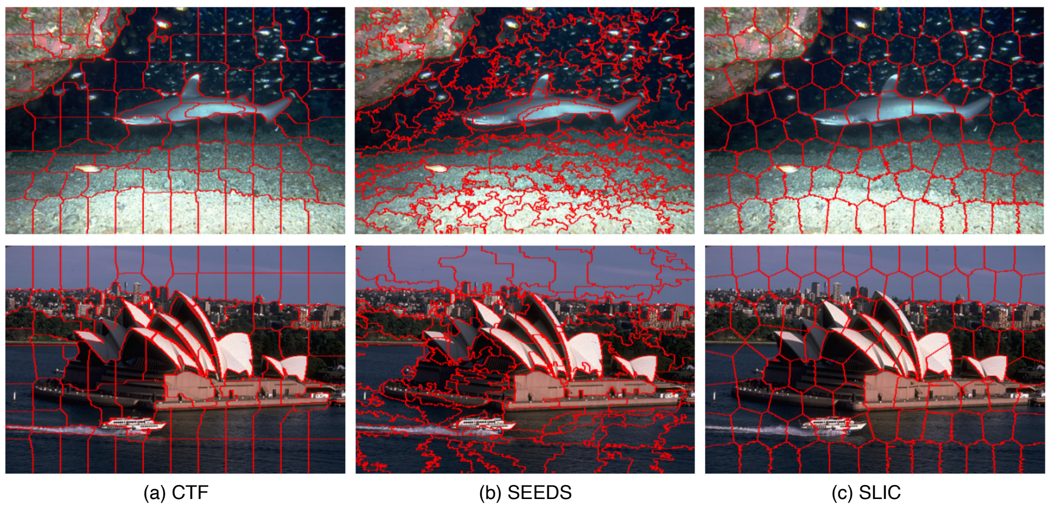


Figure 3.7: Superpixel algorithms segments images into a collection of pixels called "superpixels". Ideally, pixels within such superpixels should share visual characteristics such as color, texture. Another desirable property is that superpixel boundaries should align with the high-contrast edges in the image. Here, we observe superpixels (boundaries of which are marked in red) obtained by using the popular superpixel algorithms - SLIC (Achanta et al., 2012), SEEDS (Van den Bergh et al., 2012), and CTF (Yao et al., 2015) algorithm. Image from Yao et al. (2015).

3.3 Superpixels

Superpixels are perceptual grouping of pixels that share common characteristics (Ren and Malik, 2003), e.g., color, texture. Superpixels partitions images into a tractable set of segments, which in turn can be used as computation units for high level computer vision tasks such as object detection (Shu et al., 2013; Yan et al., 2015), object proposal generation (Pont-Tuset et al., 2017; Rantalankila et al., 2014) and semantic segmentation (Gould et al., 2008; Cadena and Košecká, 2014) to name a few. Superpixels are computationally efficient; with superpixels, it is possible to speed up such higher-level tasks significantly by orders of tens or thousands without sacrificing the performance. Typically, the input to superpixel algorithms is the desired number of superpixels N. There exists a wide variety of superpixel algorithms which can be distinguished from one another by the objective functions they minimize and the optimization algorithm (Achanta et al., 2012; Stutz et al., 2018).

Simple Linear Iterative Clustering or SLIC is a popular superpixel generation algorithm (Achanta et al., 2012) which clusters pixels based on their color similarity (in CIE-LAB color space) and spatial proximity. SLIC starts from a regular grid

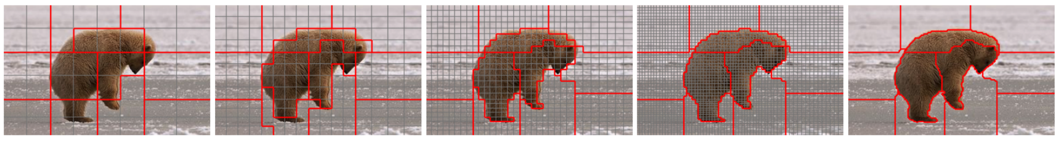


Figure 3.8: CTF (Yao et al., 2015) starts by performing boundary-level at the coarsest level (left) and then proceeds in a coarse-to-fine manner with the finest updates being applied at the pixel level (right). Image from Yao et al. (2015).

of centers (avoiding initialization of centers on edges) and performs a k-means style optimization. SLIC iteratively repeats the process of associating pixels with the nearest cluster centers and updating the cluster centers at each iteration; the process repeats till convergence. Examples of superpixels generated using SLIC are shown in Fig. 3.7. Other popular superpixel algorithms include SEEDS (Yao et al., 2015) and CTF (Yao et al., 2015). Both the framework of SEEDS (Van den Bergh et al., 2012) and CTF (Yao et al., 2015) use a coarse-to-fine energy update strategy where boundary-level updates start at the coarsest level (larger block size) and then proceeds to the finest level (pixels) iteratively (Fig. 3.8).

In this dissertation, we make use of SLIC and CTF superpixels for content-aware encoding of the user-clicks (Chapter. 4). The superpixels are generated in an unsupervised manner (Achanta et al., 2012; Yao et al., 2015). We avail the in-built MATLAB implementation of SLIC and the public implementation of CTF provided by Yao et al. (2015)[1]. We use the default parameters of the individual algorithms and *only* provide the number of superpixels as input.

3.4 Object Proposals

Object proposal generation is an image pre-processing technique which provides a collection of high-quality, class-independent object hypothesis (Carreira and Sminchisescu, 2011; Endres and Hoiem, 2013; Pont-Tuset et al., 2017). The generated proposals, in turn, can be used as input to object detection and instance segmentation methods and avoid the need for exhaustive sliding window search conventionally used in such methods. The goal for such algorithms to find a small set of object hypotheses that captures all objects in the scene.

Here, we briefly outline the framework of Multiscale Combinatorial Grouping or MCG (Pont-Tuset et al., 2017) which proposes class-independent object proposals. MCG initializes by segmenting the image (at different resolutions) into image regions. These regions are indexed by their place in the segmentation hierarchy.

[1]CTF implementation: `https://bitbucket.org/mboben/spixel`

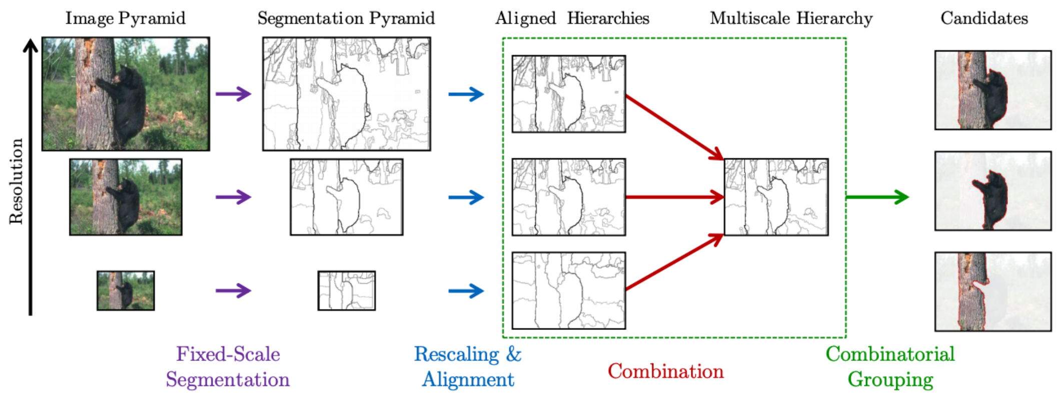


Figure 3.9: Multiscale Combinatorial Grouping (Pont-Tuset et al., 2017) (MCG) performs hierarchical segmentation at different resolutions of the input image. The generated hierarchies are aligned (via boundary snapping) and combined to generate a single multiscale segmentation hierarchy. The contour values (the higher the value, the darker it is) in the intermediate segmented images can be interpreted as the measure of contrast of that particular contour. Finally, candidate object proposals are generated by exploring the combinatorial space of the generated regions. We make use of such object proposal candidates in our user-click encodings. Image from Pont-Tuset et al. (2017).

At the finest level these image segments are superpixels; regions from the coarse level are unions of region from the finer levels. At the finest level, the superpixels are constructed on the basis of low-level features such as brightness, color, and texture difference (Martin et al., 2004), contours (Xiaofeng and Bo, 2012), and edges (Dollár and Zitnick, 2013). However, it is unlikely that such segments obtained using low-level features are able to capture complex objects in its entirety. Accordingly, to capture complete objects, MCG creates a set of object hypothesis by combinatorially merging image regions from different segmentation hierarchies.

For generation of our content-aware user click encodings (Chapter. 4), we use the pre-computed object proposals for Pascal VOC 2012 (Everingham et al., 2010) and MS COCO (Lin et al., 2014) made publicly available by the authors of MCG (Pont-Tuset et al., 2017)[2]. For obtaining the object proposals for the GrabCut (Rother et al., 2004) and the Berkeley (McGuinness and O'connor, 2010) dataset, we use the publicly available implementation of MCG (Pont-Tuset et al., 2017)[3] algorithm on the *'accurate'* setting. MCG returns, on an average, 500-1000 superpixels or regions for each image.

[2]https://www2.eecs.berkeley.edu/Research/Projects/CS/vision/grouping/mcg
[3]https://github.com/jponttuset/mcg

3.5 Evaluation Metrics

Fully automated semantic segmentation algorithms in literature (Long et al., 2015; Liang-Chieh et al., 2015; Chen et al., 2018b,c) are typically evaluated using the mean intersection over union (mIoU). First, the intersection over union (IoU) is computed for each semantic class; mIoU is then computed by the taking the average over IoUs for all the semantic classes. Semantic segmentation methods are conventionally compared using the mIoU *w.r.t* the ground truth masks of an withheld test-set (Everingham et al., 2010); The higher the mIoU, the better is the segmentation method (Chen et al., 2018c).

More formally, for an image, let y_{ij} be the number of pixels from class i that were predicted as class j and let the total number of pixels belonging to class i be given by $t_i = \sum_j y_{ij}$. Then the mIoU is given by (Long et al., 2015),

$$\text{mIoU} = \frac{1}{n_C} \frac{\sum_i y_{ii}}{(t_i + \sum_j y_{ji} - y_{ii})} \tag{3.6}$$

Any prediction errors from such automated methods can be corrected as a separate post-processing method; typically, the segmentation algorithm does not play any role beyond the prediction (Long et al., 2015; Chen et al., 2018b,c). Different from such automated segmentation methods, in interactive segmentation frameworks (Xu et al., 2016), users can add a succession of 'positive' and 'negative' clicks to improve the generated output mask by removing the false negatives and false positives, respectively. These user cues, at each step, are processed by the segmentation algorithm, which then generates an updated prediction based on the user-provided cues.

Existing click-based *deep* interactive frameworks are commonly evaluated using two metrics (Xu et al., 2016; Liew et al., 2017; Mahadevan et al., 2018; Li et al., 2018; Jang and Kim, 2019; Lin et al., 2020) - **mIoU%@clicks** and **clicks@mIoU%**. The first metric, **mIoU%@clicks**, compares the average mIoU over all the object instances in the dataset after each round of clicking. This metric quantifies the responsiveness of interactive frameworks to new user clicks. As more and more clicks are added, the underlying segmentation algorithm should respond to the user-provided cues and improve the predicted mask.

clicks@mIoU% denotes the average number of clicks required to reach a desired mIoU across all instances of the dataset. This metric primarily evaluates the suitability of a framework for an annotation task where high-quality segments are desired for each instance in the scene preferably with a low number of clicks. On the other hand, this fixed mIoU threshold is 90% for the simpler datasets, such as, GrabCut (Rother et al., 2004) and Berkeley (McGuinness and O'connor, 2010) and 85% for the more challenging Pascal VOC 2012 (Everingham et al., 2010) and MS COCO (Lin et al., 2014) datasets. Following (Xu et al., 2016), the maximum number of clicks for an instance is limited to 20 (Xu et al., 2016).

Both automated segmentation methods and interactive segmentation methods are evaluated on datasets with dense per-pixel annotations. Different to semantic segmentation methods, majority of the existing interactive frameworks operate in a class-agnostic manner. So, unlike semantic segmentation methods, class labels are not required during training and evaluation of interactive frameworks. Conventionally, interactive frameworks are trained using the ground truth masks of individual instances from the *train* set of large-scale datasets such as Pascal VOC 2012 (Everingham et al., 2010) and MS COCO (Lin et al., 2014). Such datasets contain annotated instances in the order of tens to hundreds of thousands. To demonstrate the generalization ability of interactive frameworks, they are evaluated on datasets with instances from previously unseen classes. These typically include the GrabCut (Rother et al., 2004), Berkeley (McGuinness and O'connor, 2010), and the MS COCO (Lin et al., 2014) dataset, provided instances from these datasets were not used during training.

GrabCut (Rother et al., 2004) and **Berkeley** (McGuinness and O'connor, 2010) are small-scale datasets with around 50 to 100 foreground objects. The images from GrabCut, typically, consist of a single foreground object, with a very distinctive appearance. Compared to GrabCut, the instances in the Berkeley dataset are more challenging with heavily textured and low contrast background. Both the datasets are popular for benchmarking interactive segmentation methods. **Pascal VOC 2012** (Everingham et al., 2010) consists of 1464 training and 1449 validation images across 20 object classes; majority of the images contain multiple foreground objects. For training, conventionally, 1464 images plus the additional instance annotations from **SBD** (Hariharan et al., 2011) are considered, which results in around $20,000$ instances. It is a very popular dataset for benchmarking interactive segmentation methods as well as semantic segmentation methods (Chen et al., 2018c). **MS COCO** (Lin et al., 2014) is another large-scale image segmentation dataset with instances from 80 different object categories, 20 of which are common with Pascal VOC 2012. Existing interactive frameworks, typically, do not report their performance on all the validation instances of MS COCO; instead these frameworks report the performance over 800 instances that are randomly sampled with 10 instances coming from each object class.

Chapter 4

Content-Aware Guidance Maps

This chapter includes the content of the following publication:

- Soumajit Majumder and Angela Yao. "Content-aware multi-level guidance for interactive instance segmentation." *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

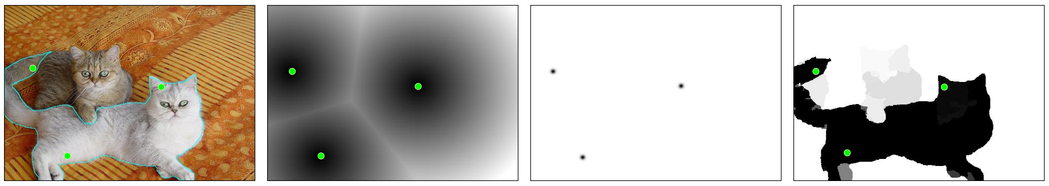


Figure 4.1: Existing interactive instance segmentation (Xu et al., 2016; Liew et al., 2017; Li et al., 2018; Hu et al., 2019) techniques do not utilize any image information when generating *guidance* maps (second column). In contrast, our proposed technique exploits image structures such as superpixels and object proposals, allowing us to generate more informative guidance maps (first column, bottom row).

In interactive instance segmentation, users give feedback to iteratively refine segmentation masks. The user-provided clicks are transformed into guidance maps which provide the network with necessary cues on the whereabouts of the object of interest. Guidance maps used in current systems are purely distance-based and are either too localized or non-informative. We propose a novel transformation of user clicks to generate content-aware guidance maps that leverage the hierarchical structural information present in an image. Using our guidance maps, even the most basic FCNs are able to outperform existing approaches that require state-of-the-art segmentation networks pre-trained on large scale segmentation datasets. We demonstrate the effectiveness of our proposed transformation strategy through comprehensive experimentation in which we significantly raise state-of-the-art on four standard interactive segmentation benchmarks.

4.1 Introduction

Interactive object selection and segmentation allows users to interactively select objects of interest down to the pixel level by providing inputs such as clicks, scribbles, or bounding boxes. The segmented results are useful for downstream applications such as image/video editing (Li et al., 2004; Benard and Gygli, 2017), image-based medical diagnosis (Wang et al., 2018a,b), human-machine collaborative annotation (Andriluka et al., 2018), etc. GrabCut (Rother et al., 2004) is a pioneering example of interactive segmentation which segments objects from a user-provided bounding box by iteratively updating a colour-based Gaussian mixture model. Other methods include Graph Cuts (Boykov and Jolly, 2001), Random Walk (Grady et al., 2005) and GeoS (Criminisi et al., 2008) though more recent methods (Xu et al., 2016; Liew et al., 2017; Mahadevan et al., 2018) approach the problem with deep learning architectures such as FCNs.

In standard, non-interactive instance segmentation (Vincent and Soille, 1991; Dai et al., 2016; Hariharan et al., 2014, 2015; He et al., 2017), the RGB image is given as input and segmentation masks for each object instance are predicted. In an interactive setting, however, the input consists of the RGB image as well as *'guidance'* maps based on user-provided supervision. The guidance map helps to select the specific instance to segment; when working in an iterative setting, it can also help correct errors from previous segmentations (Xu et al., 2016; Liew et al., 2017; Benard and Gygli, 2017; Mahadevan et al., 2018).

User feedback, in most deep learning-based interactive image segmentation approaches, is typically given in the form of point clicks (Xu et al., 2016; Liew et al., 2017; Mahadevan et al., 2018; Li et al., 2018; Hu et al., 2019). In majority of these works, these user-provided clicks are transformed into guidance maps using simplistic primitives, e.g. Euclidean-based distance transformation (Xu et al., 2016; Li et al., 2018) and Gaussian transformations (Benard and Gygli, 2017; Mahadevan et al., 2018; Maninis et al., 2018). Examples of such guidance maps can be found in Fig. 4.3.

We observe that existing user-click transformations are agnostic to the image content. Such existing guidance maps disregard even the most basic image consistencies present in the scene, such as colour and texture homogeneity (Achanta et al., 2012; Van den Bergh et al., 2012; Yao et al., 2015). This, of course, also precludes even more sophisticated structures such as object hypotheses, all of which can be determined in an unsupervised way (Pont-Tuset et al., 2017). Our motivation is to maximize the information which can be harnessed from user-provided clicks and, in that process, generate more meaningful guidance maps for interactive instance segmentation.

In this work, we propose a simple yet effective transformation of user-provided clicks which enables us to leverage a hierarchy of image information, starting from low-level cues such as appearance and texture, based on superpixels (Achanta et al., 2012; Yao et al., 2015), to more high-level information such as class-independent object hypotheses (see Fig. 4.3). Ours is the first work to investigate the impact of guidance map generation for interactive segmentation. Our findings suggest that current Gaussian- and Euclidean distance based maps are too simple and do not fully leverage structures present in the image. A second and common drawback of current distance-based guidance maps is that they fail to account for the scale of the object during interaction. Object scale has a direct impact on the network performance when it comes to classification (Papandreou et al., 2015) or segmentation (Noh et al., 2015). Gaussian- and Euclidean distance maps are primarily used for localizing the user clicks and do not account for the object scale. Our algorithm roughly estimates the object scale based on the user-provided clicks and refines the guidance maps accordingly.

Our proposed approach is extremely flexible in that the generated guidance map can be paired with any method which accepts guidance as a new input channel (Xu et al., 2016; Liew et al., 2017; Benard and Gygli, 2017; Mahadevan et al., 2018; Hu et al., 2019). We demonstrate via experimentation that providing content-aware guidance by leveraging the structured information in an image leads to a significant improvement in performance when compared to the existing state-of-the-art, all the while using a simple, off-the-shelf, CNN architecture. The key contributions of our work are as follows:

- We propose a novel transformation of user-provided clicks which generates guidance maps by leveraging hierarchical information present in a scene.
- We propose a framework which can account for the scale of an object and generate the guidance map accordingly in a click-based user feedback scheme.
- We perform a systematic study of the impact of guidance maps on the interactive segmentation performance when generated based on features at different levels of the image hierarchy.
- We achieve state-of-the-art performance on four segmentation benchmarks; the GrabCut (Rother et al., 2004), Berkeley (McGuinness and O'connor, 2010), Pascal VOC 2012 (Everingham et al., 2010) and MS COCO (Lin et al., 2014) datasets. Our proposed method significantly reduces the amount of user interaction required for accurate segmentation and uses the fewest number of average clicks per instance.

4.2 Related Works

Segmenting objects interactively using clicks, scribbles, or bounding boxes has always been a problem of interest in computer vision research, as it can solve some quality problems faced by fully-automated segmentation methods. Early variants of interactive image segmentation methods, such as the parametric active contour model (Kass et al., 1988) and intelligent scissors (Mortensen and Barrett, 1995) mainly considered boundary properties when performing segmentation; as a result they tend to fare poorly on weak edges. More recent methods are based on graph cuts (Boykov and Jolly, 2001; Rother et al., 2004; Vezhnevets and Konouchine, 2005; Li et al., 2004), geodesics (Bai and Sapiro, 2009; Criminisi et al., 2008), and or a combination of the two (Gulshan et al., 2010; Price et al., 2010). However, all these algorithms try to estimate the foreground/background distributions from low-level features such as color and texture, which are unfortunately insufficient in several instances, e.g., in images with similar foreground and background appearances, intricate textures, and poor illumination.

As with many other areas of computer vision, deep learning-based methods have become popular also in interactive segmentation in the past few years. In the initial work of Xu et al. (2016), user-provided clicks are converted to Euclidean distance transform maps which are concatenated with the color channels and fed as input to a FCN (Long et al., 2015). Clicks are then added iteratively based on the errors of the previous prediction. On arrival of each new click, the Euclidean distance transform maps are updated and inference is performed. The process is repeated until a satisfactory result is obtained. Subsequent works have focused primarily on making extensions with newer CNN architectures (Mahadevan et al., 2018; Benard and Gygli, 2017) and iterative training procedures (Mahadevan et al., 2018; Liew et al., 2017). In the majority of these works, user guidance has been provided in the form of point clicks (Xu et al., 2016; Mahadevan et al., 2018; Liew et al., 2017; Maninis et al., 2018; Li et al., 2018) which are then transformed into a Euclidean-based distance map (Xu et al., 2016; Li et al., 2018). One observation made in (Benard and Gygli, 2017; Mahadevan et al., 2018; Maninis et al., 2018) was that encoding the clicks as Gaussians led to some performance improvement because it localizes the clicks better (Mahadevan et al., 2018) and can encode both positive and negative click in a single channel (Benard and Gygli, 2017).

In Chen et al. (2018a), the authors explore the use of superpixels to generate the guidance map. However, in contrast to Chen et al. (2018a) which uses superpixels to maintain computational efficiency wrt. to their graph optimization, our guidance maps uses superpixels to leverage the local similarities contained within it. This is a general principle that we carry across image structures of varying levels for encoding user inputs. For the most part, there has been little attention paid to how user inputs should be incorporated as guidance; the main focus in interactive segmentation has been dedicated towards the training procedure and network architectures.

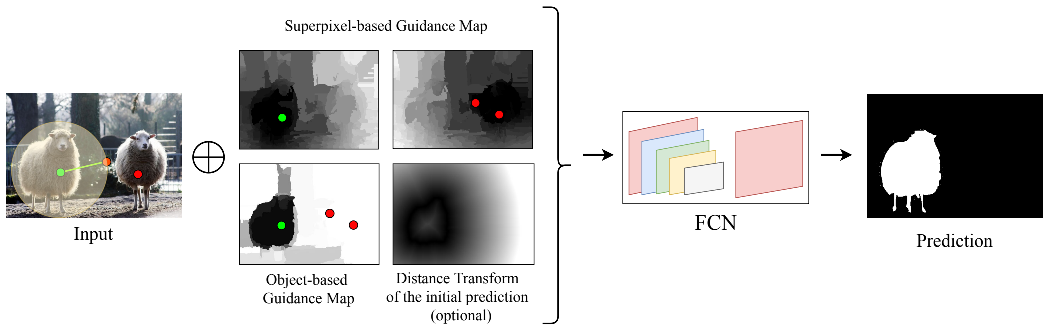

Superpixel-based Guidance Map
Input
Object-based Guidance Map
Distance Transform of the initial prediction (optional)
FCN
Prediction

Figure 4.2: Outline. Given an input image and user interactions, we transform the positive and negative clicks (denoted by the green and red dots respectively) into three separate channels (2 channel superpixel-based and 1 object proposal-based guidance map), which are concatenated (denoted as $\oplus$) with the 3-channel image input and is fed to our network. Additionally, we concatenate the euclidean distance transform of the predicted mask from the previous iteration as our final non-color channel. The solid green line indicates our estimate of the object scale based on the initial pair of positive and negative click. The desired output is the ground truth map of the selected object.

4.3 Proposed Approach

We follow previous interactive frameworks (Xu et al., 2016; Liew et al., 2017; Mahadevan et al., 2018) in which a user can provide both 'positive' and 'negative' clicks to indicate foreground and background or other objects respectively (as shown in Fig. 4.2). We denote the set of click positions as $\{\boldsymbol{p}_0, \boldsymbol{p}_1\}$ with subscripts 0 and 1 to denote positive and negative clicks respectively. To date, guidance maps have been generated by as a function of the distance between each pixel of the image grid to the point of interaction. More formally, for each pixel position $\boldsymbol{p}$ on the image grid, the pair of distance-based guidance maps for positive and negative clicks can be computed as

$$\mathcal{G}_0^d(\boldsymbol{p}) = \min_{\boldsymbol{c}\in\{\boldsymbol{p}_0\}} d(\boldsymbol{p}, \boldsymbol{c}) \quad \text{and} \quad \mathcal{G}_1^d(\boldsymbol{p}) = \min_{\boldsymbol{c}\in\{\boldsymbol{p}_1\}} d(\boldsymbol{p}, \boldsymbol{c}). \tag{4.1}$$

In the case of Euclidean guidance maps (Xu et al., 2016), the function $d(\cdot,\cdot)$ is simply the Euclidean distance; for the Gaussian guidance maps, $d(\cdot,\cdot)$ is the value of a Gaussian with a standard deviation of 10 pixels that is centered on the click (Mahadevan et al., 2018; Benard and Gygli, 2017). For the Gaussians, the min operator is replaced with the max operator. However, such guidance is image-agnostic and assumes that each pixel in the scene is independent. Our proposed approach eschews this assumption and proposes the generation of multiple guidance maps which align with both low-level and high-level image structures present in the scene. We represent low-level structures with superpixels and high-level ones with region-based object proposals and describe how we generate guidance maps from these structures in Sections 4.4 and 4.5.

4.4 Superpixel-based guidance map

We first consider a form of guidance based on non-overlapping regions; in our implementation, we use superpixels. Superpixels group together locally similarly coloured pixels while respecting object boundaries (Achanta et al., 2012) and were the standard working unit of pre-CNN -based (*classical*) segmentation algorithms (Papazoglou and Ferrari, 2013; Faktor and Irani, 2014). Previous works have shown that most, if not all, pixels in a superpixel belong to the same category (He et al., 2006; Papazoglou and Ferrari, 2013; Faktor and Irani, 2014). Based on this observation, we propagate user-provided clicks which are marked on single pixels to the entire superpixel. We then assign guidance values to each of the other superpixels in the scene based on the minimum Euclidean distance from the centroid of each superpixel to the centroid of a user-selected superpixel. One can think of the guidance as a discretized version of Eq. 4.1 based on low-level image structures.

More formally, let $\{\mathcal{S}\}$ represent the set of superpixels from an image and $f_{SP}(\boldsymbol{p})$ be a function which maps each pixel location $\boldsymbol{p}$ in the image to the corresponding superpixel in $\{\mathcal{S}\}$. We further define a positive and negative superpixel set based on the positive and negative clicks, i.e. $\{\boldsymbol{s}_0 = f_{SP}(\boldsymbol{p}_0)\}$ and $\{\boldsymbol{s}_1 = f_{SP}(\boldsymbol{p}_1)\}$ respectively. Similar to the distance-based guidance maps in Eq. 4.1, we generate a pair of guidance maps. However, rather than treating each pixel individually, we propagate the distances between superpixel centers to all pixels within each superpixel, i.e.

$$\mathcal{G}_t^{\text{sp}}(\boldsymbol{p}) = \min_{s \in \{\boldsymbol{s}_t\}} d_c\left(s, f_{SP}(\boldsymbol{p})\right), \quad \text{where} \quad t = \{0, 1\}, \tag{4.2}$$

and $d_c(s_i, s_j)$ is the Euclidean distance between the centers s_i^c and s_j^c of superpixels s_i and s_j respectively, where $s_i^c = (\sum_i x_i/|s_i|, \sum_i y_i/|s_i|)$ where $|s_i|$ denotes the number of pixels within s_i. For consistency across training images, the guidance maps values are scaled between $[0, 255]$. When the user provides no clicks, all pixel values are set to 255. Examples guidance maps are shown in the second and third column of Fig. 4.3 respectively.

4.5 Object-based guidance map

Superpixels can be grouped together perceptually into category-independent object proposals. We also generate guidance maps from higher-level image structures,

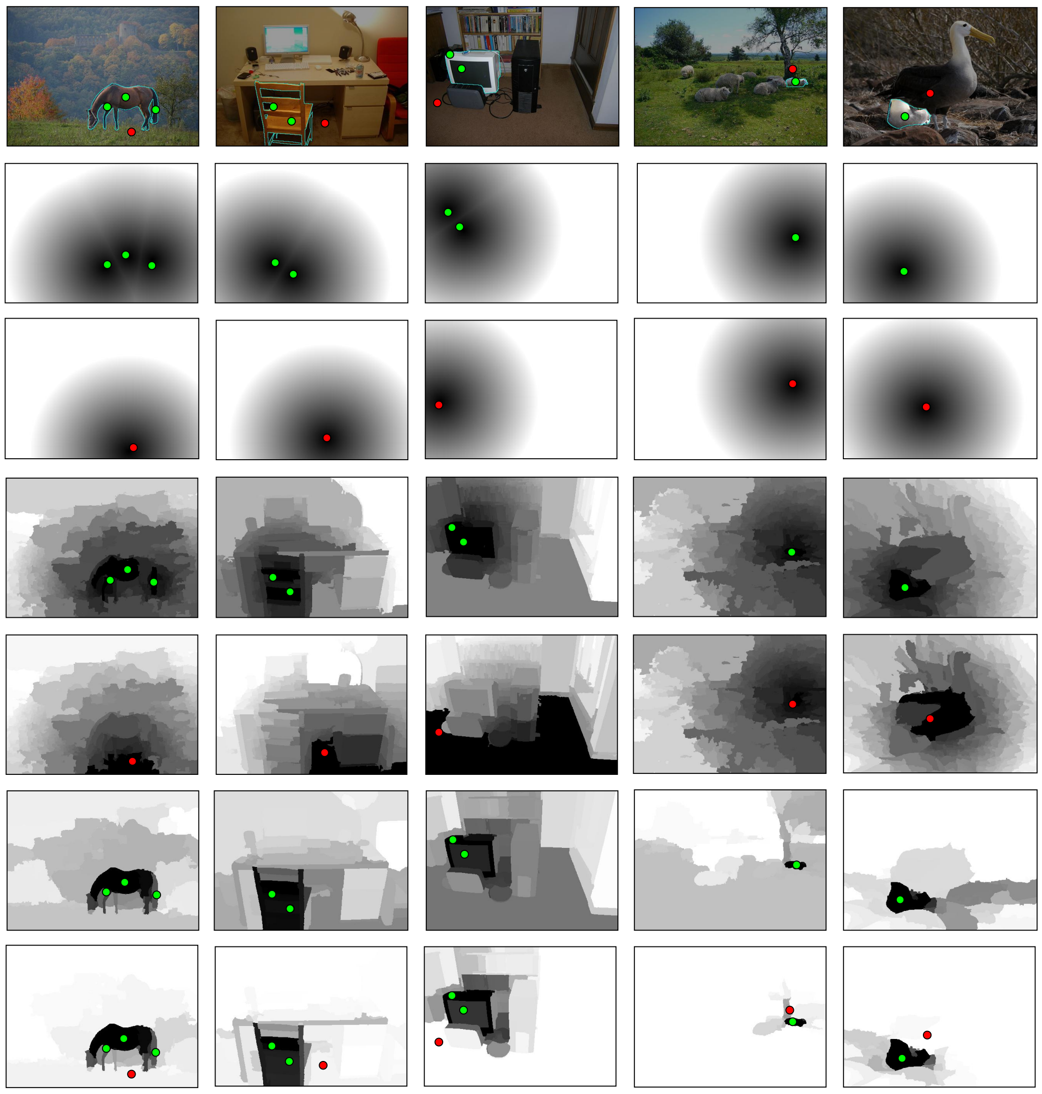


Figure 4.3: Example of content-aware guidance maps. We transform the user-provided positive (shown as green dots) and negative (shown as red dots) clicks into guidance maps for the instance segmentation network (columns 2 to 5). The second and third column correspond to the positive and negative Euclidean distance transformation based guidance map respectively (Xu et al., 2016; Liew et al., 2017). Rows 4 and 5 correspond to our proposed positive and negative superpixel based guidance map respectively. Examples of the proposed object based guidance map and the scale-aware guidance map are shown in rows 6 and 7 respectively. For the clarity of visualization, we inverted the values of the object-based guidance map and the scale-aware guidance map (Best viewed in color).

specifically region-based object proposals (Arbelaez et al., 2011; Uijlings et al., 2013; Krähenbühl and Koltun, 2014; Maninis et al., 2016; Pont-Tuset et al., 2017). Such proposals have been used in the past as weak supervision for semantic segmentation (Dai et al., 2015; Khoreva et al., 2017) and allow us to incorporate a weak object-related prior to the guidance map, even if the instance is not explicitly specified by the user-provided clicks. To do so, we begin with a set of object proposals (Pont-Tuset et al., 2017), which have positive clicks its pixel support. For each pixel in the guidance map, we count the number of proposals from this set to which the pixel belongs. Pixels belonging to same object proposals are more likely to belong in the same object category and the number of proposals to which pixels belong incorporates a co-occurrence prior with respect to the current positive clicks.

More formally, let $\{\mathcal{L}_p\}$ be the set of object proposals for an image with support of pixel location $\boldsymbol{p}$. The object-based guidance map can be generated as follows:

$$\mathcal{G}^{\text{o}}(\boldsymbol{p}) = \sum_{\boldsymbol{p'} \in \{\boldsymbol{p}_0\}} \sum_{\mathcal{L} \in \{\mathcal{L}_{p'}\}} \mathbf{1}[\boldsymbol{p} \subset \mathcal{L}] \tag{4.3}$$

where $\mathbf{1}[\boldsymbol{p} \subset \mathcal{L}]$ is an indicator function which returns 1 if object proposal $\mathcal{L}$ has in its support or contains pixel $\boldsymbol{p}$. Similar to the superpixel-base guidance map, the object-based guidance is also re-scaled to $[0, 255]$. In the absence of user-provided clicks, all pixels are set to 0. Examples are shown in the fourth column of Fig. 4.3.

4.6 Scale-aware guidance

Within an image, object instances can exhibit a large variation in their spatial extent (Singh and Davis, 2018). While deep CNNs are known for their ability to handle objects at different scales (Liang-Chieh et al., 2015), specifying the scale explicitly leads to an improvement in performance (Papandreou et al., 2015). Interactive instance segmentation methods (Maninis et al., 2018) which isolate the object tend to have a superior performance. For segmenting object instances, it is thus desirable to construct guidance maps which exhibit spatial extents consistent with the object.

A common limitation of most click-based interactive approaches is that the provided guidance is non-informative about scaling of the intended object instance. The commonly used forms of guidance are either too localized (Mahadevan et al., 2018) (guidance map values are clipped to 0 at a distance of 20 pixels from the clicks) or non-informative (Xu et al., 2016).

Suppose now that we have some rough estimate of an object's scale in pixels, either in width or length. A convenient way to make our guidance maps scale-aware is to incorporate contributions of superpixels and object proposals which are in agreement with this scale. More specifically, we can apply this to the superpixel guidance map by truncating distances exceeding some factor f of our scale measure s, i.e.,

$$\mathcal{G}_t^{\text{sp-sc}}(\boldsymbol{p}) = \min\left[\mathcal{G}_t^{\text{sp}}(\boldsymbol{p}), f s\right]. \tag{4.4}$$

We can apply similar constraints to the object-proposal based guidance by considering only the proposals within an accepted size range bounded by tolerance factors f_1 and f_2:

$$\mathcal{G}^{\text{o-sc}}(\boldsymbol{p}) = \sum_{\boldsymbol{p'} \in \{\boldsymbol{p}_0\}} \sum_{\mathcal{L} \in \{\mathcal{L}_{p'}\}} \mathbf{1}[\boldsymbol{p} \subset \mathcal{L}] \cdot \mathbf{1}[f_1 \leq |\mathcal{L}|/s^2 \leq f_2]. \tag{4.5}$$

4.6.1 Simulating user interactions

Even when selecting the same object instance, it is unlikely that different users will provide the same interactions inputs. For the model to fully capture expected behaviour across different users, one would need significant amounts of interaction training data. Rather than obtaining these clicks from actual users for training, we simply simulate user clicks and generate guidance maps accordingly.

We follow the sampling strategies proposed in Xu et al. (2016). For each object instance, we sample N_{pos} positive clicks within the object maintaining a distance d_1^{in} pixels from the object boundary and d_2^{in} pixels from each other. For negative clicks, we test the first two of the three sampling strategies outlined in Xu et al. (2016), one in which N_{neg}^1 clicks are sampled randomly from the background, ensuring a distance of d_1^{out} pixels away from the object boundary and d_2^{out} pixels from each other and one in which N_{neg}^2 clicks on each of the negative objects (objects not of interest).

The above click-sampling strategy helps the network to understand notions such as negative objects and background but cannot train the network to identify and correct errors made during the prediction (Mahadevan et al., 2018). To this end, we randomly sample N_{iter} clicks based on the segmentation errors. After an initial prediction is obtained, positive or negative clicks are randomly sampled from the error. Existing set of clicks are then replaced with the newly sampled clicks with a probability of 0.3. To mimic a typical user's behavior (Mahadevan et al., 2018), the error-correction clicks are placed closest to the center of the largest misclassified (false positive or false negative) region.

To estimate the scale measure s, we reserve the first two clicks, one positive and one negative, and assume that the Euclidean distance between the two is a roughly proportional measure; f, f_1 and f_2 are then set accordingly.

4.7 Experimental Validation

4.7.1 Datasets and Evaluation

We apply our proposed guidance maps and evaluate the resulting instance segmentations on four publicly available datasets: PASCAL VOC 2012 (Everingham et al., 2010), GrabCut (Rother et al., 2004), Berkeley (McGuinness and O'connor, 2010), and MS COCO (Lin et al., 2014).

Evaluation. Fully automated instance segmentation is usually evaluated with mean intersection over union (mIoU) between the ground truth and predicted segmentation mask. Interactive instance segmentation is differently evaluated because a user can always add more positive and negative clicks to improve the segmentation and thereby increase the mIoU. As such, the established way of evaluating an interactive system is according to the number of clicks required for each object instance to achieve a fixed mIoU. Similar to existing frameworks (Xu et al., 2016; Liew et al., 2017; Mahadevan et al., 2018; Benard and Gygli, 2017), we limit the maximum number of clicks per instance to 20. Unlike Xu et al. (2016); Liew et al. (2017), we do not apply any post-processing with a conditional random field and directly use the segmentation output from the FCN.

4.7.2 Implementation Details

Training. As our base segmentation network, we adopt the FCN-8s (Long et al., 2015) pre-trained on Pascal VOC 2012 dataset (Everingham et al., 2010) as provided by MatConvNet (Vedaldi and Lenc, 2015). The output layer is replaced with a two-class softmax layer to produce binary segmentations of the specified object instance. We fine-tune the network on the 1464 training images with instance-level segmentation masks of Pascal VOC 2012 segmentation dataset (Everingham et al., 2010) together with the 10582 masks of SBD (Hariharan et al., 2011). We further augment the training samples with random scaling and flipping operations. We use zero initialization for the extra channels of the first convolutional layer (`conv1_1`). Following (Xu et al., 2016), we fine-tune first the stride-32 FCN variant and then the stride-16 and stride-8 variants. The network is trained to minimize the average binary cross-entropy loss. For optimization, we use a learning rate of 0.01 and stochastic gradient descent (SGD) with Nesterov momentum with the default value of 0.9 is used.

Click Sampling. We generate training images with a variety of click numbers and locations; sometimes, clicks end up being sampled from the same superpixel, which reduces training data variation. To prevent this and also make the network more robust to the click number and location for training, we sample randomly from the following hyperparameters rather than fixing them to single values: $N_{pos} =$

$\{2,3,4,5\}$, $N^1_{neg} = \{5,10\}$, $N^2_{neg} = \{3,5\}$, $d^{in}_1 = \{15,20,40\}$, $d^{in}_2 = \{7,10,20\}$, $d^{out}_1 = \{15,40,60\}$, $d^{out}_2 = \{10,15,25\}$. The randomness in the number of clicks and their relative distances prevents the network from over-fitting during training.

Guidance Dropout. Since the FCNs are pre-trained on Pascal VOC 2012, we expect the network to return a good initial prediction for images with object instances from one of its 20 classes. Thus, during training, when the network receives images without any instance ambiguity (i.e. an image with single object), we zero the guidance maps (value of 0 for object guidance map and 255 for the superpixel based guidance map) with a probability of 0.2 to encourage good segmentations without any guidance. We further increase robustness by resetting the positive or negative superpixel-based guidance with a probability of 0.4.

Interaction Loop. During evaluation, a user provides positive and negative clicks sequentially to segment the object of interest. After each click is added, the guidance maps are recomputed; in addition the distance transform of predicted mask from the previous iteration is provided as an extra channel (Mahadevan et al., 2018). The newly generated guidance map is concatenated with the image and given as input to the FCN-8s network which produces an updated segmentation map.

Superpixels and Object Proposals. We use the implementation provided in Pont-Tuset et al. (2017) for generating superpixels; on average, each frame has 500−1000 superpixels. For comparison, we also try other superpixelling variants, e.g. SLIC (Achanta et al., 2012) and CTF (Yao et al., 2015). Although several other object proposal algorithms exist (Carreira and Sminchisescu, 2011; Uijlings et al., 2013; Pont-Tuset et al., 2017), we use only MCG (Pont-Tuset et al., 2017) as it has been shown to have higher quality proposals. The final stage of MCG returns a ranking which we disregard. We use the pre-computed object proposals for Pascal VOC 2012 and MS COCO provided by the authors of Pont-Tuset et al. (2017). For GrabCut and Berkeley, we run MCG (Pont-Tuset et al., 2017) on the *'accurate'* setting to obtain our set of object proposals[1].

4.7.3 Impact of Structure-Based Guidance.

We begin by looking at the impact of superpixel based guidance. As a baseline, we compare with Xu et al. (2016), which uses a standard Euclidean distance-based guidance as given in Eq. 4.1 (second and third column of Fig. 4.3). Similar to Xu et al. (2016), we concatenate our positive and negative superpixel-based guidance maps with the three color channels and feed it as an input to the FCN-8s (Long et al., 2015). We use the superpixels computed using MCG (Pont-Tuset et al.,

[1]https://github.com/jponttuset/mcg

2017). For a fair comparison, we train our network non-iteratively, i.e., during training, we do not generate click samples based on the error in the prediction and do not append the distance transform of the current predicted mask as an extra channel. Looking at Table. 4.1, we see that our superpixel based guidance maps significantly reduce the number of clicks required to reach the standard mIoU threshold.

The object-based guidance provides the network with a weak localization prior of the object of interest. Adding the object-based guidance with the superpixel based guidance leads to further improvements in performance (see third row of Table. 4.1). The impact is more prominent for datasets with a single distinct foreground object (e.g. 9.3% and 14% relative improvement for the Berkeley and GrabCut dataset). Finally, by making the feedback iterative, i.e. based on previous segmentation errors, we can further reduce the number of clicks. Overall, our structure-based guidance maps can reduce the number of clicks by 35% to 47% and unequivocally proves that having structural information in the guidance map is highly beneficial.

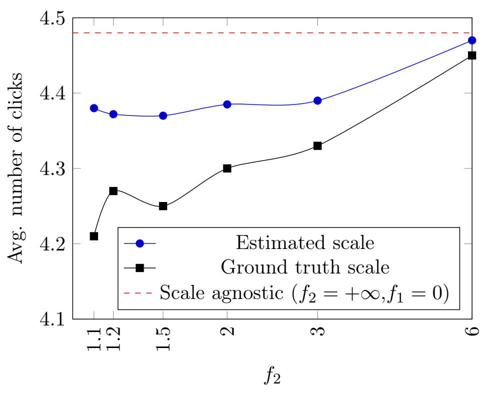
	GrabCut @90%	Berkeley @90%	VOC 2012 @85%
Euclidean (Xu et al., 2016)	6.04	8.65	6.88
SP	4.44	6.67	4.23
SP+Obj.	3.82	6.05	4.02
SP+Obj.+Iter	**3.58**	**5.60**	**3.62**

Table 4.1: Clicks required for different types of guidance. Guidance maps leveraging structural information require significantly less clicks than Euclidean distance-based guidance. *SP* refers to the superpixel guidance maps and *Obj* refers to the obect based guidance map and *Iter* refers to iterative training.

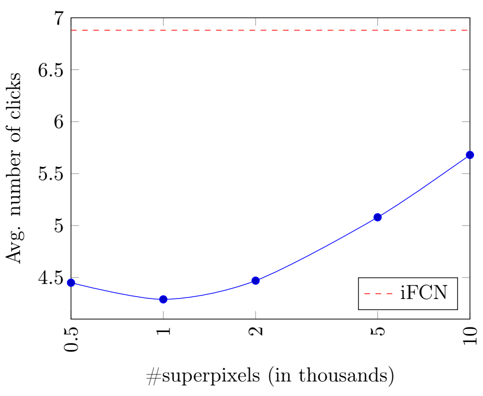
4.7.4 Impact of Scale-Aware Guidance

Due to fixed-size receptive field, FCNs experience difficulty when segmenting small objects (Noh et al., 2015). The benefits of our scale-aware guidance map is most pronounced for segmenting small objects; for larger objects ($\geq 32 \times 32$ pixels), it does not seem to have that much of an impact. To highlight the impact of our guidance on small object instances, we pick the subset of 621 objects from Pascal VOC 2012 (Everingham et al., 2010) which are smaller than 32×32; objects smaller than this size are harder to identify (Singh and Davis, 2018).

In the scale agnostic setting, we consider all object proposals which has the click in its pixel support for generating the object-based guidance map, i.e. (as shown in Equation. 4.3; note that this is equivalent to having $f_1 = 0, f_2 = \infty$). Since the lower bound on scale has little effect, we set $f_1 = 0$. Looking at the average number



(a) Scale-Aware Guidance



(b) Number of superpixels

Figure 4.4: (a) Scale-Aware Guidance. The figure shows the average number of clicks required for segmenting small object instances (smaller than 32 × 32 pixels (Singh and Davis, 2018)) for varying degrees of tolerance till which we accept object proposals for generating our guidance map based on our estimated object scale and the ground truth object scale (computed as the square root of the number of pixels in the object mask). (b) Number of superpixels. The figure shows the average number of clicks required for segmenting object instances in Pascal VOC 2012 *val* set for different number of superpixels.

of clicks required per instance to reach 85% mIoU for the subset of small objects (see Fig. 4.4 (a)), we find that having a soft scale estimate improves the network performance when it comes to segmenting smaller objects. This is primarily because the guidance map disregards object proposals which are not consistent in scale and can degrade the network performance by inducing a misleading co-occurrence prior.

When the scale s is based on ground truth (as the square root of the number of pixels in the mask foreground, see black curve in Fig. 4.4 (a)), the average clicks required per instance is consistently lower than the scale-agnostic case, even when as we relax f_2 up to 6, i.e. allowing for object proposals which are 6 times larger than the actual object scale. Estimating the scale from the clicks is of course much less accurate than when it is take from the ground truth masks (compare blue curve vs black curve in Fig. 4.4 (a)). Nevertheless, even with such a coarse estimate, we find improvements in the number of clicks required as compared to the scale-agnostic scenario (compare red dashed line in Fig. 4.4 (a)). Given the first pair of positive and negative clicks, our estimated object scale is $\sqrt{\pi}d$ where d is the euclidean distance between the positive and the negative click. In our experiments, we observed that our estimated scale varies between 50-300% from the ground truth scale. In comparison to a scale-agnostic setting, over the Pascal VOC 2012 *val* set, we observe an improvement of 0.1 clicks (a relative improvement of 2%) on the small objects subset and an improvement of 0.032 clicks per instance for objects larger than 32×32 pixels. Segmenting small objects with CNNs can be problematic (Noh et al., 2015); we observed similar difficulties in preliminary experiments. For objects smaller than 32×32 pixels from Pascal VOC 2012 *val* set, we require an average of 4.33 clicks which is significantly higher than our dataset average of 3.62 clicks.

4.7.5 Superpixels

Type of Superpixels. To study the impact of the superpixeling algorithm, we consider the two variants SLIC (Achanta et al., 2012) and CTF (Yao et al., 2015) and use only the superpixel based guidance map. On an average, MCG (Pont-Tuset et al., 2017) generates $500 - 1000$ superpixels for each image in its default setting. For a fair comparison, we generate 500 and 1000 superpixels using SLIC and CTF. We observe that using 1000 SLIC superpixels results in performance similar to the MCG (see Table. 4.2). However, irrespective of the superpixeling method, we found an overall improvement when the guidance maps are generated based on superpixels instead of pixel-based distances.

#superpixels	SLIC (Achanta et al., 2012)	CTF (Yao et al., 2015)	MCG (Pont-Tuset et al., 2017)
500	4.45	4.82	4.23
1000	4.29	4.58	

Table 4.2: Choice of superpixel algorithm. In this setting we only consider the superpixel-based guidance maps and report the average number of clicks required to attain a mIoU of 85% for all instances in the Pascal VOC 2012 *val* set.

Number of Superpixels. We now study the impact of the number of superpixels. For this, we consider only the superpixel-based map as guidance and use SLIC (Achanta et al., 2012) as the superpixel algorithm. In the extreme case, all superpixels will have one pixel in its support and the guidance map degenerates to the Euclidean distance transform commonly used in existing interactive methods (Xu et al., 2016; Li et al., 2018). We use the reported results in iFCN (Xu et al., 2016) on Pascal VOC 2012 *val* set as our degenerate case (as shown by the red curve in Fig. 4.4 (b)). In addition to the reported results for 500 and 1000 superpixels on Pascal VOC 2012 *val* set, we generate $2000, 5000$ and 10000 superpixels using SLIC (Achanta et al., 2012), as shown in Fig. 4.4(b). We notice an initial gain in performance, but with increase in the number of superpixels, the performance drops as our network requires more and more clicks to segment the object of interest. As the number of superpixels increase, the benefits of local structure based grouping is lost as each superpixel is segmented into similar and redundant superpixels.

Method	Base Network	GrabCut @90%	Berkeley @90%	VOC-2012 @85%	COCO-20 @85%	COCO-60 @85%
$\text{GC}_{\text{ICCV'01}}$	-	15.06	11.10	14.33	18.67	17.80
$\text{RW}_{\text{MICCAI'05}}$	-	11.37	12.30	14.02	13.91	11.53
$\text{GM}_{\text{ICCV'09}}$	-	14.75	12.44	15.96	17.32	14.86
$\text{ESC}_{\text{CVPR'10}}$	-	11.79	8.52	12.11	13.90	11.63
$\text{GSC}_{\text{CVPR'10}}$	-	11.73	8.38	12.57	14.37	12.45
$\text{iFCN}_{\text{CVPR'16}}$	FCN-8s (Long et al., 2015)	6.04	8.65	6.88	8.31	7.82
$\text{RIS}_{\text{ICCV'17}}$	Deeplab (Liang-Chieh et al., 2015)	5.00	6.03	5.12	5.98	6.44
$\text{VOS-Wild}_{\text{arXiv'17}}$	Deeplabv3+ (Chen et al., 2018c)	3.80	-	5.60	-	-
$\text{ITIS}_{\text{BMVC'18}}$	Deeplabv3+ (Chen et al., 2018c)	5.60	-	3.80	-	-
$\text{LD}_{\text{CVPR'18}}$	CAN (Yu and Koltun, 2016)	4.79	-	-	12.45	12.45
$\text{Two-Stream}_{\text{NN'19}}$	VGG-16 (Simonyan and Zisserman, 2014)	3.76	6.49	4.58	9.62	9.62
Ours	FCN-8s (Long et al., 2015)	**3.58**	**5.60**	**3.62**	**5.40**	**6.10**

Table 4.3: The average number of clicks required to achieve a particular mIoU on four public benchmarks by classical interactive frameworks - GC (Boykov and Jolly, 2001), RW (Grady et al., 2005), GM (Bai and Sapiro, 2009), ESC (Gulshan et al., 2010), and GSC (Gulshan et al., 2010); and deep learning-based interactive frameworks - iFCN (Xu et al., 2016), RIS (Liew et al., 2017), VOS-Wild (Benard and Gygli, 2017), ITIS (Mahadevan et al., 2018), LD (Li et al., 2018), and Two-Stream (Hu et al., 2019). The best results are indicated in **bold**. We observe that even with an earlier segmentation backbone (Long et al., 2015), our proposed framework is able to outperform existing deep learning based approaches using state-of-the-art segmentation models (Chen et al., 2018b,c). COCO-20 and COCO-60 refers to the instances from 20 overlapping and 60 non-overlapping *w.r.t* Pascal VOC 2012 object categories respectively.

4.7.6 Comparison to State of the Art

We compare the average number of clicks required to reach some required mIoU (see Table. 4.3) against other methods reported in the literature. The methods vary in the base segmentation network from the basic FCNs to the highly sophisticated Deeplabv3+ and also make use of additional CRF post-processing. We achieve the lowest number of clicks required for all datasets across the board, again proving the benefits of applying guidance maps based on existing image structures. We report results for our best trained SP+Obj+Iter network. To reach the mIoU threshold of 90% on GrabCut and Berkeley, our full model needs the fewest number of clicks as shown in Table. 4.3 with a relative improvement of 5.79% and 7.13% over the current benchmark. For Pascal VOC 2012 *val* set, we observe a relative improvement of 4.7%. For MS COCO, we observe a larger improvement for the 20 seen categories from Pascal VOC 2012, as our networks were trained heavily on these object categories. Overall, we achieve an improvement of 9.7% and 5.28% over the 20 seen and 60 unseen object categories. We note that such an improvement is achieved despite the fact that our base network is the most primitive of the methods compared, i.e. an FCN-8s, in comparison to the others who use more complex (Deeplabv3+) network architectures with much deeper (ResNet-101) backbones. It should be noted that FCTSFN (Hu et al., 2019) and IIS-LD (Li et al., 2018) report their result over all the 80 classes of MS COCO and not separately for 20 seen and 60 unseen classes. We also compare our approach to that of Chen et al. (2018a) which targets images with only a single foreground object. To be comparable, we consider only our results with a single positive (foreground) click. We find that for the GrabCut and Berkeley dataset, our mIoU is higher by 4% and 8% respectively.

4.8 Qualitative Results

Zero Clicks. We show via some qualitative examples, the benefits of having the guidance dropout. In several instances, our network is able to produce high quality masks without any user guidance (as shown in Fig. 4.5).

Multiple Clicks. In Fig. 4.6, we show some examples where undesired objects and background was removed with only a few clicks resulting in a suitable object mask.

Failure Cases. We show some examples from Pascal VOC 2012 *val* set, where our network is unable to generate object masks with $\geq 85\%$ mIoU and exhausts the 20 click budget (see Fig. 4.7). These failure cases are representative of the problems faced by CNNs while segmenting objects from images such as, small objects (Noh et al., 2015), occlusion, motion blur and objects with very fine structures. In general, we observed that our network had difficulty in handling three object classes from Pascal VOC 2012 - *chair, bicycle* and *potted plant*. This stems from the inability of

CNNs to produce very fine segmentations, most likely due to the loss of resolution from downsampling in the encoder.

4.9 Discussion and Conclusion

In this work, we investigated the impact of the guidance maps for interactive object segmentation. Conventional methods use distance transform based approaches for generating guidance maps which disregard the inherent image structure. We proposed a scale aware guidance map generated using hierarchical image information which leads to significant reduction in the average number of clicks required to obtain a desirable object mask. During experimentation, we observed that the object instances within the datasets varied greatly in difficulty. For instance, on Pascal VOC 2012, the base network, without *any* user guidance, is able to meet the $\geq 85\%$ mIoU criteria for 433 of the 697 instances. Similarly observations were made for GrabCut ($\geq 90\%$ mIoU, 13 out of 50) and Berkeley ($\geq 90\%$ mIoU, 15 out of 100). On the other hand, we encountered instances where our algorithm repeatedly exhausted the 20 click budget regardless of sampled click locations and iterative feedback based on prediction errors. This is especially true for objects with very fine detailing, such as spokes in bicycle wheels, partially occluded chairs, etc. Based on these two extreme cases, we conclude that interactive segmentation is perhaps not so relevant for single object instances featuring prominently at the center of the scene and should feature more challenging scenarios. On the other hand, we need to design better algorithms which can handle objects that are not contiguous in region, i.e. has holes and are able to handle scenarios of occlusion. Depending on the target application, dedicated base architectures may be necessary to efficiently handle these cases.

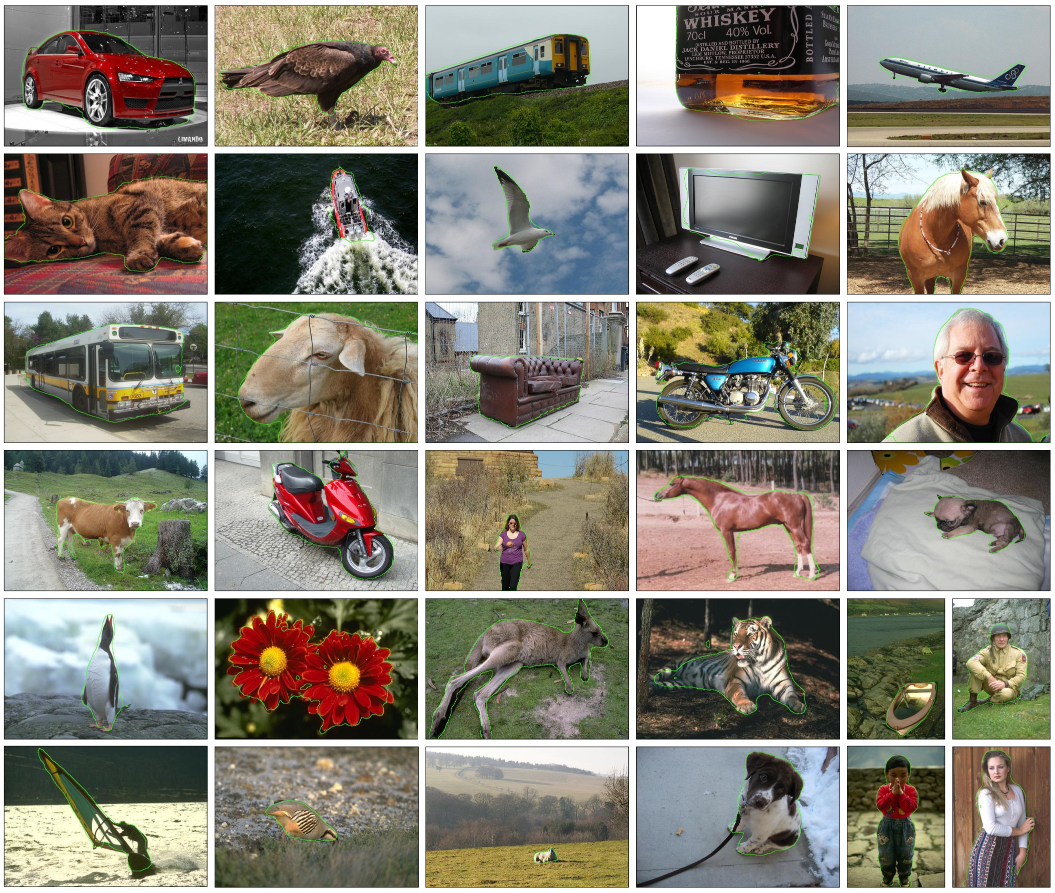


Figure 4.5: Zero Clicks. Examples of high-quality object masks generated without any user guidance. Generated object boundaries are shown in green (Best viewed in color).

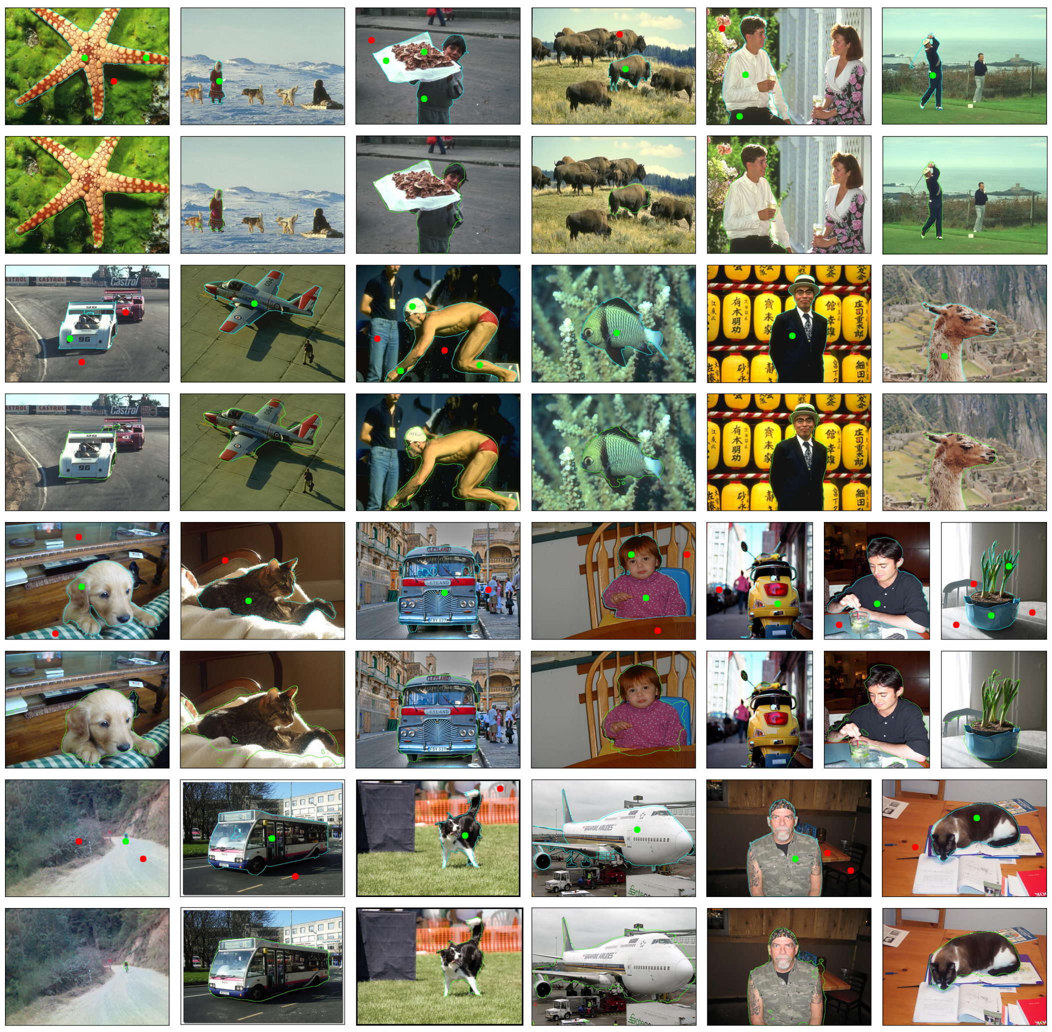


Figure 4.6: Multi Clicks. With a few clicks, background and undesired objects were removed from the final prediction mask. Green dots indicate positive click, red dots indicate negative click. Ground truth object boundaries are shown in cyan and predicted object boundaries are shown in green (Best viewed in color).

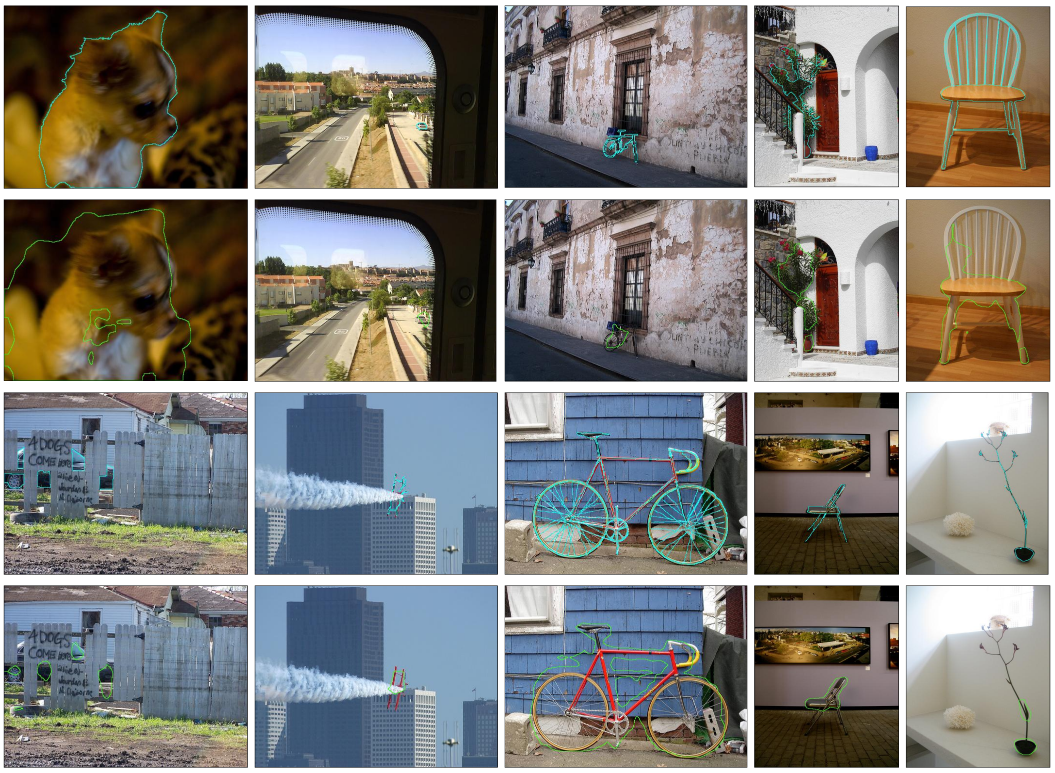


Figure 4.7: Failure Cases. Examples of failure cases. Ground truth object boundaries are shown in cyan. Generated object boundaries from the predicted mask are shown in green. In the *dog* example, the network has difficulty distinguishing the fur from the background. For the *car* example, it is either too small (1st row, 2nd column) or too occluded (2nd row, 1st column). For the *bicycle, chair* and *potted plant* example, the error in prediction is due to the inability of the network in handling very fine structures. (Best viewed in color).

Chapter 5

Localized Interactive Instance Segmentation

This chapter includes the content of the following publication:

- Soumajit Majumder and Angela Yao. “Localized Interactive Instance Segmentation.” *DAGM German Conference on Pattern Recognition (GCPR)*, 2019.

In current interactive instance segmentation works, the user is granted a free hand when providing clicks to segment an object; clicks are allowed on background pixels and other object instances far from the target object. This form of interaction is highly inconsistent with the end goal of efficiently isolating objects of interest. In our work, we propose a clicking scheme wherein user interactions are restricted to the proximity of the object. In addition, we propose a novel transformation of the user-provided clicks to generate a weak localization prior on the object which is consistent with image structures such as edges, textures etc. We demonstrate the effectiveness of our proposed clicking scheme and localization strategy through detailed experimentation in which we raise state-of-the-art on several standard interactive segmentation benchmarks.

5.1 Introduction

Interactive object selection or interactive instance segmentation allows users to select objects of interest down to a pixel level by providing inputs such as clicks, scribbles and bounding boxes. The selected results are useful for downstream applications such as image/video editing (Benard and Gygli, 2017; Li et al., 2004), medical diagnosis (Wang et al., 2018a,b), image annotation tools (Andriluka et al., 2018; Benenson et al., 2019) etc. GrabCut (Rother et al., 2004) is a pioneering example of interactive segmentation; other notable methods include Lazy Snapping (Li et al., 2004) and Random Walk (Grady et al., 2005).

More recent methods (Hu et al., 2019; Liew et al., 2017; Majumder and Yao, 2019a; Maninis et al., 2018; Xu et al., 2016) have approached the problem with deep learning architectures such as fully convolutional networks (FCNs). In deep interactive segmentation, the input consists of the RGB image as well as *‘guidance’* maps based on user-provided supervision. Users give ‘*positive*’ clicks on the object of interest and '*negative*' clicks on the background or other objects in the scene. The guidance map helps the network to focus on the object instance to segment; in an iterative setting, it helps to correct errors from previous segmentations (Benard and Gygli, 2017; Liew et al., 2017; Mahadevan et al., 2018; Majumder and Yao, 2019a; Xu et al., 2016). Typically, such guidance maps are generated via fixed rules and are not visible to the end user; only the image and intermediate and end segmentation results are visible to the interacting user.

For deep interactive segmentation, research efforts have predominantly been limited to introducing new architectures (Benard and Gygli, 2017; Mahadevan et al., 2018) and more sophisticated training procedures (Liew et al., 2017; Mahadevan et al., 2018). Yet minimizing user interaction and maintaining high quality segmentation requires a fine interplay between good specification of user interactions and careful leveraging of the provided inputs. Previous methods have ignored these aspects by allowing users to freely provide inputs (Benard and Gygli, 2017; Mahadevan et al., 2018; Xu et al., 2016) in any order and at any location in the scene.

In addition, the guidance generated from the clicks are primitive and agnostic to structures present in the image (Benard and Gygli, 2017; Mahadevan et al., 2018; Maninis et al., 2018). In fact, the number and type of clicks to give, as well as how to encode user clicks are open research questions with enormous impact on the performance of the interactive system. For example, (Majumder and Yao, 2019a) showed that with improved click encodings, a simple base segmentation network such as the FCN-8s (Long et al., 2015) can outperform methods (Benard and Gygli, 2017; Mahadevan et al., 2018; Maninis et al., 2018) that use much deeper and stronger base networks such as ResNet-101 (He et al., 2016). In this work, we follow along this line of work in looking at how to cleverly specify and leverage user clicks to improve interactive instance segmentation.

In this work, our interest is in directing user clicks to weakly constrain the area of interest for interactive segmentation. Limiting the spatial extent is advantageous both for the network and the user, i.e. it tells the network which area to focus on for learning and also gives some indication of object scale; it also directs the user clicks to ambiguous locations which will most benefit from guidance.

Directing user clicks to specify the location may seem like an obvious way for interaction but few works on interactive segmentation have done so to date. Instead, they favour hard constraints enacted by directly cropping out the bounding boxes derived from user-given inputs (Maninis et al., 2018) or object detections (Xu et al., 2017b). This hard crop relies on highly specific user inputs such as extreme points (Maninis et al., 2018) which may slow down the user interaction, or having pre-trained object detectors for the object classes of interest for segmentation (Xu et al., 2017b). We favour a simple approach, where we ask users to first roughly localize objects with the first two interactions, e.g. on the two opposite corners in a bounding box, or clicking at the center of the object and one outside the object boundary. We propose using these first two interactions or clicks as the initial form of interaction. Ensuing corrective positive and negative clicks are constrained to be outside and within the enclosing boundary.

In addition, we propose a new transformation scheme for the user-provided clicks which provides a weak localization prior on the object of interest and is consistent with low-level structures such as edges, textures, etc in the scene. Unlike (Xu et al., 2017b), this prior is generated without using class-specific bounding box detections. With the arrival of newer clicks, this proposed transformation gradually refines the localization prior. Our proposed approach can deal naturally with several types of guidance modalities, including superpixel-based guidances (Majumder and Yao, 2019a) and bounding box type guidances (Xu et al., 2017b). Our key contributions are as follows:

- We propose a simple yet efficient clicking scheme which focuses the user's attention to the object of interest and its vicinity.
- We propose a novel transformation of the user clicks which provides a weak localization prior on the object; with the arrival of new user clicks the generated guidance map gradually refines to the object boundary.
- Our proposed approach achieves state-of-the-art performance on three interactive image segmentation benchmarks including the challenging MS COCO (Lin et al., 2014) dataset; like other competing state-of-the-art methods in literature, through simulation of user clicks, we significantly reduce the amount of user input required to generate accurate segmentation.

5.2 Related Works

The development of automated semantic and instance segmentation frameworks is a rapidly growing area in computer vision (Chen et al., 2018c; Yu et al., 2018; Zhang et al., 2018). Accompanying this line of work is *interactive* segmentation - where users give clicks, scribbles, or bounding boxes to adjust and improve the outputs of these fully automated methods. Early interactive image segmentation approaches include parametric active contours, snakes (Kass et al., 1988) and intelligent scissors (Mortensen and Barrett, 1995). Since these methods focus primarily on boundary properties, they suffer when edge evidence is weak. More recent methods are based on graph cuts (Boykov and Jolly, 2001; Li et al., 2004; Rother et al., 2004; Vezhnevets and Konouchine, 2005), geodesics (Bai and Sapiro, 2009; Criminisi et al., 2008), or a combination of both (Gulshan et al., 2010; Price et al., 2010). However, since these methods try to separate foreground and background solely based on low-level features such as colour and texture, they are not robust and fare poorly when segmenting images with similar foreground and background appearances, intricate textures, and poor lighting.

Recently, deep convolutional neural networks (CNNs) have been incorporated into interactive segmentation frameworks. The initial work of Xu et al. (2016) uses Euclidean distance maps to represent user-provided positive and negative clicks which are then concatenated with the original colour image and provided as input to a fully convolutional network (Long et al., 2015). Following works have focused primarily on making extensions with newer CNN architectures (Benard and Gygli, 2017; Mahadevan et al., 2018) and iterative training procedures (Liew et al., 2017; Mahadevan et al., 2018). Instead of training with fixed user clicks as input (Xu et al., 2016), iterative training algorithms (Liew et al., 2017; Mahadevan et al., 2018) progressively add clicks based on the error of the network predictions.

In the majority of interactive segmentation frameworks, user guidance has been provided in the form of point-wise clicks (Hu et al., 2019; Li et al., 2018; Liew et al., 2017; Mahadevan et al., 2018; Maninis et al., 2018; Xu et al., 2016) which are then transformed into a Euclidean distance map (Hu et al., 2019; Li et al., 2018; Xu et al., 2016). One observation made in (Benard and Gygli, 2017; Mahadevan et al., 2018; Maninis et al., 2018) was that encoding the clicks as Gaussians led to performance improvement because it localizes the clicks better (Mahadevan et al., 2018) and can encode both positive and negative click in a single channel (Benard and Gygli, 2017). A more recent work (Benenson et al., 2019) observed encoding user clicks as small binary disks to be more effective than Gaussian and the Euclidean encoding.

Different to (Benard and Gygli, 2017; Mahadevan et al., 2018; Xu et al., 2016; Maninis et al., 2018; Benenson et al., 2019; Liew et al., 2017), we use guidance maps which are consistent with the low-level image structures. Additionally, we propose a superpixel box guidance map which provides weak localization cues to the network. This is similar in spirit to (Benenson et al., 2019; Maninis et al., 2018; Xu et al., 2017b) in which object bounding boxes are cropped out from extreme points specified by the user (Maninis et al., 2018), (loose) ground truth bounding boxes (Benenson et al., 2019) or object detections (Xu et al., 2017b). Our work relaxes the hard constraint of (Maninis et al., 2018), wherein clicks have to be placed on the four extremities of the object and on the object boundary. Furthermore, unlike (Xu et al., 2017b), our proposed superpixel box guidance is class-agnostic and does not require having pre-trained object detectors available.

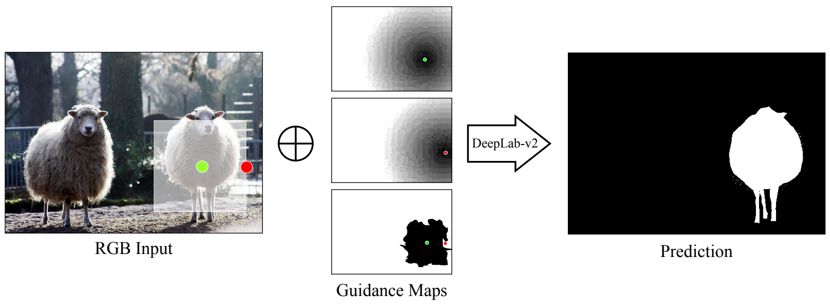


Figure 5.1: Outline. Given an image and user clicks, we transform the positive and negative clicks (denoted by the green and red circles respectively) into three separate channels - 2 channel superpixel-based (middle column, rows 1-2) and 1 superpixel-box (middle column, row 3) guidance map. These are concatenated (denoted by $\oplus$) with the 3-channel image input and is fed to our base segmentation network.

5.3 Proposed Method

We adopt the common approach for interactive segmentation that has been used in previous deep learning-based frameworks (Benard and Gygli, 2017; Liew et al., 2017; Majumder and Yao, 2019a; Mahadevan et al., 2018; Xu et al., 2016). The user provides inputs on the original RGB image in the form of *'positive'* and *'negative'* clicks to indicate foreground and background respectively. The clicks are then encoded into guidance maps via transformations (Section 5.3.2 and Section 5.3.3).

Typically, pixel values on the guidance map are a function of the pixel distance on the image grid to the points of interaction (see Fig. 5.2). This includes Euclidean (Hu et al., 2019; Xu et al., 2016) and Gaussian guidance maps (Benard and Gygli, 2017; Mahadevan et al., 2018; Maninis et al., 2018). However, such guidance maps are generated in an image-agnostic manner with the assumption that pixels in an image are independent of one another. Alternative variants take image structures such as superpixels (Chen et al., 2018a; Majumder and Yao, 2019a) and region-based object proposals (Majumder and Yao, 2019a) into consideration for generating guidance maps. Guidance maps are then concatenated as additional channels to the input image and passed through the network (Xu et al., 2016; Liew et al., 2017; Maninis et al., 2018; Chen et al., 2018a) (see Fig. 5.1).

5.3.1 Interaction Loop

In previous works (Benard and Gygli, 2017; Hu et al., 2019; Mahadevan et al., 2018; Majumder and Yao, 2019a; Xu et al., 2016), the user has the liberty to provide clicks anywhere in the scene. This includes clicking on object instances far from the one of interest. Intuitively, a user interested in recovering an object instance from the scene would primarily fixate in the vicinity of the object of interest and focus more on delineating the object from the nearby background. Additionally, unconstrained clicks on the background and other objects fail to provide hints on the whereabouts of the object which calls for additional click sampling strategies are proposed (Xu et al., 2016; Benard and Gygli, 2017; Mahadevan et al., 2018) to ensure negative clicks encompassing the object.

We propose a simple yet intuitive interaction framework. At the onset of interaction, the user provides a click at the center of the object of interest followed by another click on a background pixel in the vicinity of the object (see Fig. 5.2). This first pair of clicks is used to generate a coarse prior on the location of the object (see Sec. 5.3.3) in the form of an enclosing box. We then restrict the locations of user subsequent inputs. More specifically, negative clicks need to be given inside the estimated bounding box, while positive clicks need to be given outside. In turn, the new positive clicks are then used to update the location prior.

Let us denote the set of positive and negative clicks as $\{\boldsymbol{c}_i^+\}$ and $\{\boldsymbol{c}_i^-\}$ respectively for $i = \{1, \cdots, n\}$ and the initial foreground and background click as $\boldsymbol{c}_0^+$ and $\boldsymbol{c}_0^-$ respectively. Based on $\{\boldsymbol{c}_0^+, \boldsymbol{c}_0^-\}$, a coarse prior $\mathcal{G}$ on the object location is generated (see Sec. 5.3.3). We then restrict the locations of user subsequent inputs. More specifically, negative clicks need to be given inside the estimated object location, while positive clicks need to be given outside. The new positive clicks are used to update the bounding box boundaries $\tilde{e}_0$ and $\tilde{e}_1$, while all additional clicks, $\boldsymbol{c}_{i\neq 0}^+$ and $\boldsymbol{c}_{i\neq 0}^-$, are used to update $\mathcal{G}$.

5.3.2 Superpixel-based Guidance Maps

Superpixels are known for their ability to group locally similar pixels (Faktor and Irani, 2014; He et al., 2006; Papazoglou and Ferrari, 2013). For our guidance maps, we consider the superpixel-based variant of (Majumder and Yao, 2019a) which outperformed approaches using Euclidean and Gaussian guidance maps. In (Majumder and Yao, 2019a), user clicks given at single pixels are propagated to entire superpixels. Guidance values of other superpixels in the scene are then given by the minimum Euclidean distance from the centroid of each superpixel to the centroid of a user-selected superpixel. Example of such superpixel-based guidance maps are shown in Fig. 5.2.

More specifically, let $\{\mathcal{Z}\}$ denote the set of superpixels constituting an image and $f_{\mathcal{Z}}^{p}$ denote a function which maps every pixel p to its corresponding superpixel. Let $\{z^+\} = f_{\mathcal{Z}}^{p}(\{\boldsymbol{c}^+\})$ and $\{z^-\} = f_{\mathcal{Z}}^{p}(\{\boldsymbol{c}^-\})$ be the set of positive and negative superpixels based on the user-provided clicks. The value of each pixel for the guidance map $\mathcal{S}^+_{\mathcal{Z}}(p)$ corresponding to the set of positive clicks $\{\boldsymbol{c}^+\}$ is given by,

$$\mathcal{S}^+_{\mathcal{Z}}(p) = \min_{z \in \{z^+\}} d_c^2(z, f^p_{\mathcal{Z}}(p)), \tag{5.1}$$

and likewise for $\mathcal{S}^-_{\mathcal{Z}}(\cdot)$ for $\{\boldsymbol{c}^-\}$. In Equation 5.1, $d_c^2(z_i, z_j)$ is the Euclidean distance between the centroids z_i^c and z_j^c of superpixels z_i and z_j respectively, where $z_i^c = (\sum_i x_i/|z_i|, \sum_i y_i/|z_i|)$ and $|z_i|$ is the number of superpixels in z_i. The values of the guidance maps are truncated to 255. Examples of such guidance maps are shown in Fig. 5.2.

We additionally experimented with guidance maps generated based on the CIE-LAB color difference between the annotated superpixels and the other superpixels as per (Chen et al., 2018a) but we did not observe any promising results.

5.3.3 Superpixel-box Guidance Maps

Cropping images to exactly contain the object of interest has been shown to improve interactive segmentation performance (Benenson et al., 2019; Maninis et al., 2018). However, such frameworks are limited by the placement of the additional clicks; the framework of Maninis et al. (2018) requires corrective clicks to be placed precisely on object boundaries. Besides, this also leads to a set of unnatural training images dominated by the object of interest. This prevents the network from learning from the background regions. Unlike (Maninis et al., 2018), we refrain from cropping the image to contain only the object of interest.

Instead, we provide, as an additional guidance channel, a weak prior on the whereabouts of the object in the scene based on the initial pair of clicks. At the onset, it behaves like a weak bounding box albeit consistent with low-level image features. With the arrival of additional clicks, it gets further refined into sloppy contours (Dutt Jain and Grauman, 2013), and provides the segmentation network with a strong cue on the location of the objects (see Fig. 5.2). Unlike object-based guidance maps (Majumder and Yao, 2019a) generated based on object proposals (Pont-Tuset et al., 2017), our proposed guidance is more flexible and adapts more quickly to the user-provided inputs.

More formally, given the first pair of positive and negative click $\boldsymbol{c}_0^+ = (x_0^+, y_0^+)$ and $\boldsymbol{c}_0^- = (x_0^-, y_0^-)$ for an image of size $w \times h$, we obtain the top-left and the bottom-right co-ordinates of the object, e_0 and e_1 respectively. Let $\{\mathcal{Z}_b\} \subset \{\mathcal{Z}\}$ be the set of superpixels which lie on or inside the spatial extent defined by e_0 and e_1. The value of each pixel p of the superpixel-box based guidance map is given by,

$$\mathcal{G}(p) = \mathbf{1}[p \subset z] \cdot \mathbf{1}[z \subset \{\mathcal{Z}_b\}] \tag{5.2}$$

where $\mathbf{1}[p \subset z]$ is an indicator function which returns 1 if pixel p lies belongs to superpixel z. $\mathbf{1}[z \subset \{\mathcal{Z}_b\}]$ returns 1 if superpixel z belongs to the set of superpixels $\{\mathcal{Z}_b\}$. For the additional set of clicks $\{\boldsymbol{c}^{+}_{i\neq 0}\}$ and $\{\boldsymbol{c}^{-}_{i\neq 0}\}$, we obtain the updated guidance map $\mathcal{G}\hat{(}p)$ as follows,

$$\{z^{+}_{i\neq 0}\} = f^{p}_{\mathcal{Z}}(\{\boldsymbol{c}^{+}_{i\neq 0}\}) \tag{5.3}$$

$$\{z^{-}_{i\neq 0}\} = f^{p}_{\mathcal{Z}}(\{\boldsymbol{c}^{-}_{i\neq 0}\}) \tag{5.4}$$

$$\{\hat{\mathcal{Z}}_b\} = \{\mathcal{Z}_b\} \cup \{z^{+}_{i\neq 0}\} \setminus \{z^{-}_{i\neq 0}\} \tag{5.5}$$

$$\mathcal{G}\hat{(}p) = \mathbf{1}[p \subset z] \cdot \mathbf{1}[z \subset \{\hat{\mathcal{Z}}_b\}] \tag{5.6}$$

5.3.4 Simulating User Interactions

To train and test our network, we simulate user interactions, as per previous works on interactive segmentation (Xu et al., 2016; Liew et al., 2017; Mahadevan et al., 2018). For simulating user interactions, we make use of the ground truth masks of Pascal VOC 2012 (Everingham et al., 2010) along with the additional masks from Semantic Boundaries Dataset (SBD) (Hariharan et al., 2011). We use the centroid of the ground truth masks as our first positive click; for concave object masks, clicks falling outside the object mask are relocated to a point within the object. We then displace the click location by 20-50 pixels randomly; we ensure that the final click location remains within the object. This is done to introduce variation in the training data; the perturbation prevents center clicks to always fall on the same superpixel during each training iteration and also better approximates true user interactions which may not perfectly localize the object center.

Next, we sample the first negative click which is at least d pixels away from the center click; in our experiments for a bounding box of height h and width w, we set d to be,

$$d = (r_1 - r_2) \cdot w + (1 + r_2) \cdot h \tag{5.7}$$

where r_1 is sampled from the uniform distribution $\mathcal{U}(0, 1)$ and r_2 is sampled from the normal distribution $\mathcal{N}(0, 1)$. We use the first pair of clicks to generate the enclosing superpixel box; we keep superpixel boxes with an intersection over union of ≥ 0.7. For simulating additional positive and negative clicks, we pick 2-5 superpixels at random from the set of superpixels lying outside the enclosing box and inside respectively.

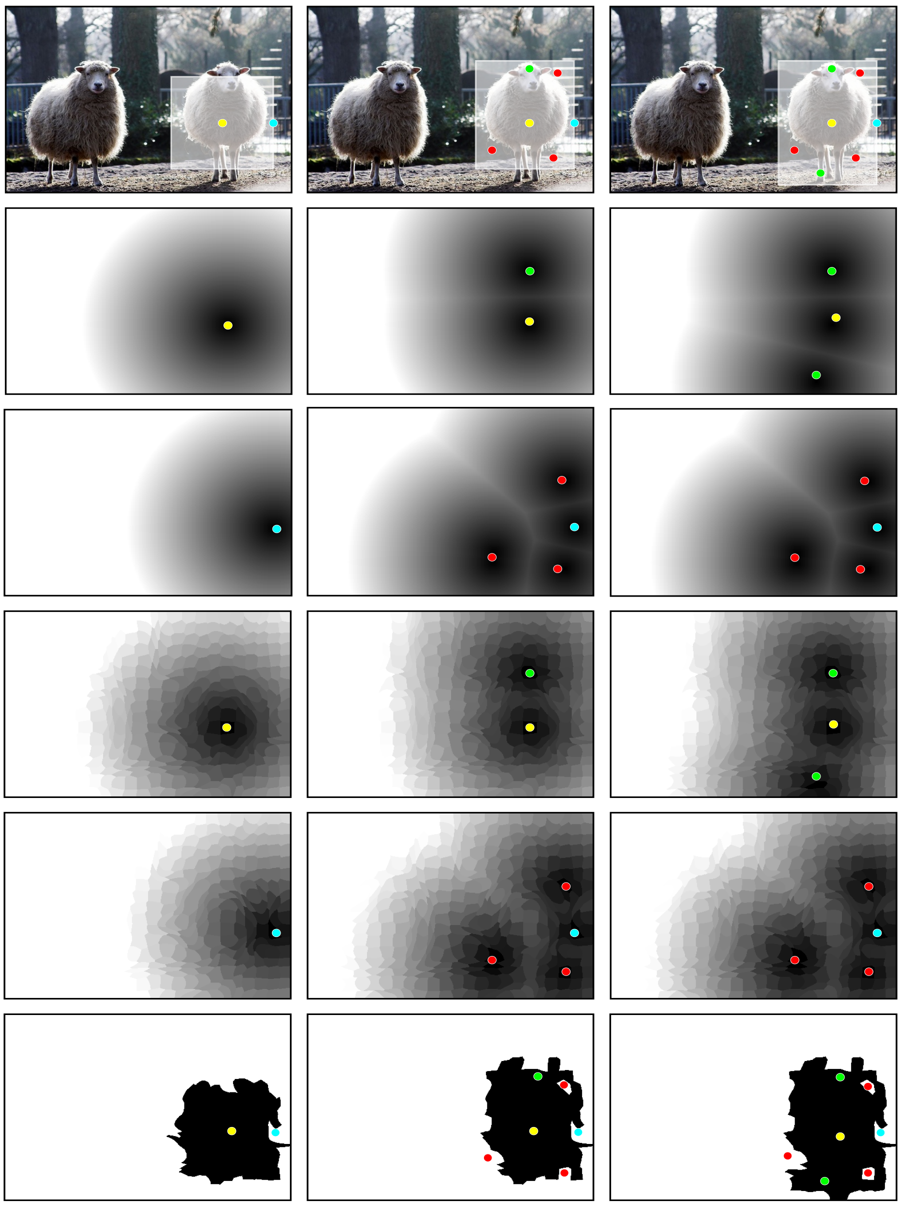


Figure 5.2: Examples of Guidance maps. At the onset of interaction, our approach receives an initial pair of enclosing clicks (denoted by yellow and blue at the center of the object and on a background pixel respectively). These clicks are transformed into guidance maps. With each round of additional clicking, the guidance maps are updated by considering the positive clicks (shown in green) and negative clicks (shown in red). Examples of user click transformations are shown in rows 2 to 6; rows 2-3: positive and negative Euclidean distance maps, rows 4-5: positive and negative superpixel-based guidance maps, row 6: the superpixel-box guidance. The values of the superpixel-box guidance are inverted for ease of visualization. The image along with the guidance maps are used as input for the segmentation network. *Note that euclidean distance maps are not used as guidance maps in our approach.*

5.4 Experimentation

5.4.1 Datasets and Evaluation

Dataset. We evaluate the performance our proposed method on five publicly available datasets used for benchmarking interactive image and video segmentation (Chen et al., 2018a; Liew et al., 2017; Majumder and Yao, 2019a; Maninis et al., 2018; Xu et al., 2016): Pascal VOC 2012 (Everingham et al., 2010), GrabCut (Rother et al., 2004), Berkeley (McGuinness and O'connor, 2010), MS COCO (Lin et al., 2014) and DAVIS 2016 (Perazzi et al., 2016). DAVIS 2016 is a dataset for video object segmentation. It consists of 50 video sequences 20 from which are in the validation set. The sequences feature a single foreground object; the pixel mask of the object is provided for all frames. For fair comparison with (Xu et al., 2016; Liew et al., 2017; Majumder and Yao, 2019a), we split the dataset into the 20 Pascal VOC 2012 categories and the 60 additional categories, and randomly sample 10 images per category for evaluation.

Evaluation. The performance of fully automated instance segmentation algorithms is usually measured by the average mean intersection over union (mIoU) between the ground truth masks and the predicted object masks. In interactive segmentation, a user can always provide more positive and negative clicks to further improve the predicted segmentation. The established way of evaluating an interactive system is based on the number of clicks required for each object instance to achieve a fixed mIoU (Xu et al., 2016). This fixed mIoU threshold is 90% for GrabCut and Berkeley and 85% for the more challenging Pascal VOC 2012 and MS COCO. Like (Xu et al., 2016; Benard and Gygli, 2017; Liew et al., 2017; Mahadevan et al., 2018; Majumder and Yao, 2019a), we threshold the maximum number of clicks per instance to 20 clicks. Unlike (Xu et al., 2016; Benard and Gygli, 2017; Liew et al., 2017), we do not apply any post-processing with a conditional random field.

5.4.2 Implementation Details

Model Architecture. As our base segmentation network, we use Deeplabv2 (Chen et al., 2018b); it consists of a ResNet-101 (He et al., 2016) backbone and a Pyramid Scene Parsing network (Zhao et al., 2017) acting as the prediction head. The output of the CNN is a probability map representing whether a pixel belongs to the object. We initialize the weights from a network Deeplabv2 model pre-trained on ImageNet (Russakovsky et al., 2015), and fine-tuned on Pascal VOC 2012 (Everingham et al., 2010) for semantic segmentation.

Training Data. We further tune the network for instance segmentation on the 1464 training images of Pascal VOC 2012 (Everingham et al., 2010) with the instance-level masks, along with the 10582 images of SBD (Hariharan et al., 2011). We further augment the training samples with random scaling, flipping and rotation operations.

Superpixels. We use SLIC (Achanta et al., 2012) as our superpixeling algorithm. We generate around 1000 superpixels on an average per image; using 1000 SLIC superpixels over 500 SLIC superpixels have been shown to improve performance (Majumder and Yao, 2019a). Generating finer superpixels ($\geq$ 2000 superpixels per image) degrades the performance as the superpixel-based guidance map degenerates to the Euclidean distance map. During evaluation on the GrabCut (Rother et al., 2004), Berkeley (McGuinness and O'connor, 2010), MS COCO (Lin et al., 2014) and DAVIS 2016 (Perazzi et al., 2016) dataset, we roughly generate 1000 SLIC superpixels (Achanta et al., 2012) for each image.

Training Details. Our network is trained to minimize a pixel-wise binary cross-entropy loss between the ground truth mask and the predicted mask. For optimization, we use stochastic gradient descent with Nesterov momentum with its default value of 0.9. The learning rate is fixed at 10^{-8} across all epochs and weight decay is $5 \cdot 10^{-4}$. A mini-batch of size 5 is used. The implementation is done in PyTorch and built on top of the implementation provided by (Maninis et al., 2018). We train our network for 50 epochs.

Guidance Dropout. Dropout can be incorporated into the guidance inputs by introducing fixed-value maps during training with some probability. Guidance dropout has been shown to be effective for interactive segmentation (Majumder and Yao, 2019a) since it encourages the base segmentation network which is trained for semantic segmentation to switch over to instance segmentation without any user interaction. Following Majumder and Yao (2019a), during training, when the network receives an image with single object, we fix the value of 255 for the superpixel-based and the superpixel-box based guidance map with a probability of 0.1 to encourage good initial segmentations in absence of clicks. Additionally to make it robust to the number of user clicks, during training, we provide guidance maps with a single positive click (at the center) and the initial positive-negative click pair with a probability of 0.1.

5.4.3 Ablation Studies

We perform an ablation study to analyze the impact of different components in our interactive instance segmentation pipeline on the final segmentation output. We use the Berkeley dataset (McGuinness and O'connor, 2010) for performing our ablation studies (see Table. 5.1). Similar to the observation in (Majumder and Yao, 2019a), using a superpixel-based guidance map leads to a significant improvement over its euclidean distance map counterpart (denoted by EU) as used in iFCN (Xu et al., 2016) (Table. 5.1, rows 1-2). We observe additional gains from adopting

EU	SP	BBox	SPBox	DT	Base Network	Berkeley @90%
✓					FCN-8s	8.65
	✓				FCN-8s	6.67
	✓				Deeplabv2	6.32
	✓	✓		✓	Deeplabv2	5.49
	✓	✓			Deeplabv2	5.26
	✓		✓		Deeplabv2	5.18

Table 5.1: Ablation Study. We report the average number of clicks need to reach 90% mIoU over all instances in the Berkeley dataset (McGuinness and O'connor, 2010) under different configurations of the guidance maps as well as segmentation models. For rows 1-2, we report the values directly from Xu et al. (2016) and Majumder and Yao (2019a) respectively.

the more recent ResNet-101 (He et al., 2016) as our backbone architecture w.r.t FCN-8s (Long et al., 2015) as used in (Majumder and Yao, 2019a) (Table. 5.1, row 3).

Next, the benefits of having a weak localization prior on the object of interest as an additional mode of guidance (Table. 5.1, rows 4-6). BBox refers to the rectilinear box drawn between corner pixel locations e_0 and e_1 generated from user clicks $\{\boldsymbol{c}_i^+\}$ and $\{\boldsymbol{c}_i^-\}$. SPBox refers to the superpixel-box guidance generated from $\{\boldsymbol{c}_i^+\}$ and $\{\boldsymbol{c}_i^-\}$ (Sec. 5.3.3, Equations 2-6). Having a weak localization prior is shown to improve results across the board; the improvement is higher when using the SPBox guidance. Additionally, we consider the distance transform (DT) of the BBox as guidance but the average number of clicks increase from 5.26 to 5.49. Throughout our experiments, we use the superpixel-based guidances and the superpixel-box guidance as our guidance maps.

5.4.4 Comparison to State of the Art

We compare the average number of clicks required to reach a required mIoU (see Table. 5.2) against existing interactive segmentation approaches. We achieve the lowest number of clicks required for the GrabCut and for the challenging MS COCO (both seen and unseen categories) datasets, proving the benefits of restricting the interaction to only the object of interest. In Fig. 5.3, we show some qualitative results from the Pascal VOC 2012 *val* set. As shown in Table. 5.2, our full model needs the fewest number of clicks to reach the required mIoU threshold of 90% on GrabCut, with a relative improvement of 3.3%. For MS COCO, we observe an improvement of 4.6% and 6.5% over the 20 seen and 60 unseen object categories respectively. We also report a relative 7.5% improvement for Pascal VOC 2012 *val* set w.r.t previous

Method	GrabCut @90%	Berkeley @90%	VOC-2012 @85%	COCO-20 @85%	COCO-60 @85%
iFCN$_{\text{CVPR'16}}$	6.04	8.65	6.88	8.31	7.82
RIS$_{\text{ICCV'17}}$	5.00	6.03	5.12	5.98	6.44
VOS-Wild$_{\text{arXiv'17}}$	3.80	-	5.60	-	-
ITIS$_{\text{BMVC'18}}$	5.60	-	3.80	-	-
LD$_{\text{CVPR'18}}$	4.79	-	-	12.45	12.45
Two-Stream$_{\text{NN'19}}$	3.76	6.49	4.58	9.62	9.62
CAG$_{\text{CVPR'19}}$	3.58	5.60	**3.62**	5.40	6.10
BRS$_{\text{CVPR'19}}$	3.60	**5.08**	-	-	-
Ours	**3.46**	5.18	3.70	**5.15**	**5.70**

Table 5.2: Average number of clicks required to achieve a fixed mIoU across state-of-the-art deep learning-based interactive frameworks - iFCN (Xu et al., 2016), RIS (Liew et al., 2017), VOS-Wild (Benard and Gygli, 2017), ITIS (Mahadevan et al., 2018), LD (Li et al., 2018), Two-Stream (Hu et al., 2019), CAG (Majumder and Yao, 2019a), LIS (Majumder and Yao, 2019b), and BRS (Jang and Kim, 2019). Best results are indicated in **bold**. COCO-20 and COCO-60 refers to the instances from 20 overlapping and 60 non-overlapping categories *w.r.t* Pascal VOC 2012 classes.

state-of-art algorithms using a fixed clicking scheme (Maninis et al., 2018). For MS COCO dataset, it should be noted that Two-Stream (Hu et al., 2019) and LD (Li et al., 2018) report their result averaged over all the 80 object categories.

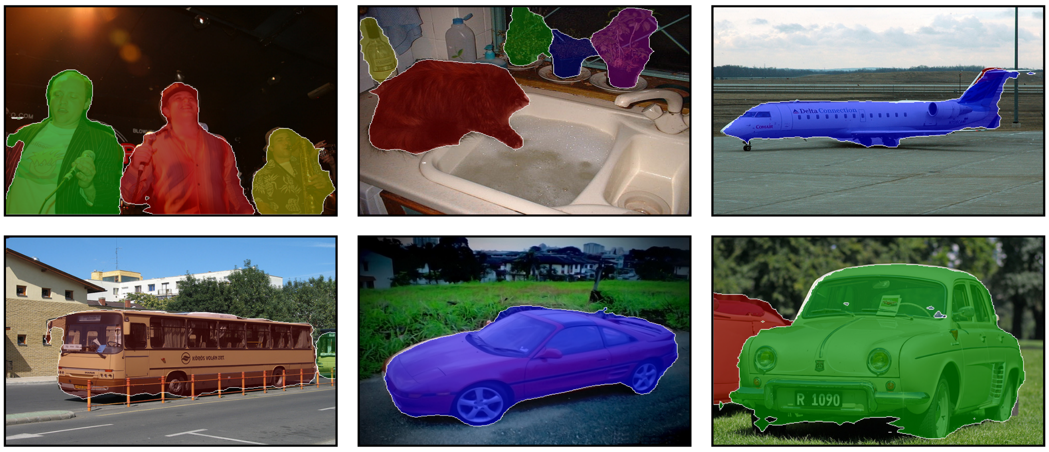


Figure 5.3: Qualitative Results. Examples of high-quality object segmentations generated on the Pascal VOC 2012 *val* set. Note that final segmentation masks might not align to object boundaries as no CRF-based post-processing was performed. For ease of visualization, we skip plotting the user-provided clicks.

5.4.5 Correcting Masks for Video Object Segmentation

Fully automated video object segmentation techniques can generate object segmentation masks of unsatisfactory quality; such masks are unsuitable for their intended downstream application. Such scenarios can benefit from interactive segmentation approaches. Given an unsatisfactory prediction, users can provide additional clicks to improve the mask. OSVOS (Caelles et al., 2017) is such a video object segmentation algorithm. It fine-tunes the segmentation network using the ground truth mask of the first frame in a video; it then segments objects in the videos in a frame-by-frame manner. Following (Benard and Gygli, 2017; Mahadevan et al., 2018), we proceed to improve the worst segmentation masks per sequence as generated by OSVOS on the DAVIS 2016 (Perazzi et al., 2016) validation set. We report the mIoU after the addition of 1, 4 and 10 clicks.

Method	OSVOS	1-click	4-clicks	10-clicks
GrabCut(Rother et al., 2004)	50.4	46.6	53.5	68.8
iFCN(Xu et al., 2016)	50.4	55.7	71.3	79.9
VOS-Wild(Benard and Gygli, 2017)	50.4	63.8	75.7	82.2
ITIS(Mahadevan et al., 2018)	50.4	67.0	77.1	82.8
Ours	50.4	**72.2**	**80.1**	**84.3**

Table 5.3: Refinement of the worst predictions from OSVOS on the DAVIS 2016 *val* set. We report the updated mIoU after the addition of 1, 4, and 10 clicks for refining the segmentations generated using OSVOS.

We initialize our enclosing area for the superpixel-box guidance map based on the initial segmentation by OSVOS. Superpixel-based guidance maps are set to a value of 255. We then provide additional positive and negative clicks to improve the mask quality which are then used to update the superpixel-based guidance maps. Our proposed algorithm reports a significant gain of over 5% in mIoU for a single click and also outperforms the reported results for 4 and 10 clicks (see Table. 5.3).

5.5 Conclusion

In this chapter, we demonstrate that limiting the extent of user interaction to only the object of interest can significantly reduce the amount of user interaction required to obtain satisfactory segmentations. Additionally, via experiments, we demonstrate the benefits of having a weak localization prior generated in the form of superpixel box guidance. Our proposed algorithm primarily faced difficulties when trying to segment occluded instances. In such cases, the superpixel box guidance overlaps significantly, making it difficult for the network to segment both the instances properly.

Chapter 6

Two-in-One Refinement for Interactive Segmentation

This chapter includes the content of the following publication:

- Soumajit Majumder, Abhinav Rai, Anshul Khurana, and Angela Yao. "Two-in-One Refinement for Interactive Segmentation." *British Machine Vision Conference (BMVC)*, 2020.

Deep convolutional neural networks are now mainstream for click-based interactive image segmentation. Most frameworks refine false negatives and false positive regions via a succession of positive and negative clicks placed centrally in these regions. In this chapter, we propose a simple yet intuitive two-in-one refinement strategy placing clicks on the boundary of the object of interest. As boundary clicks are a strong cue for extracting the object of interest and we find that they are much more effective in correcting wrong segmentation masks. In addition, we propose a boundary-aware loss that encourages segmentation masks to respect instance boundaries. We place our new refinement scheme and loss formulation within a task-specialized segmentation framework and achieve state-of-the-art performance on the standard datasets - Berkeley, Pascal VOC 2012, DAVIS 2016, and MS COCO. We exceed competing methods by $6.5\%, 9.4\%, 10.5\%$ and 2.5%, respectively.

6.1 Introduction

The goal of interactive image segmentation is to obtain an accurate pixel-level mask for an object with minimal user input. It is a fundamental processing stage for applications such as image editing (Benard and Gygli, 2017) and medical imaging analysis (Wang et al., 2018a). More recently, with the increased popularity of deep learning, the demand for mask-level annotations for segmentation tasks has risen dramatically. Manual labelling of such data is highly laborious; a single image can take as long as 1.5 hours for Cityscapes (Cordts et al., 2016). Alternatively, one can leverage the advances of automated segmentation and use human-in-the-loop frameworks (Benenson et al., 2019; Andriluka et al., 2018; Maninis et al., 2018).

The holy grail of interactive instance segmentation is to achieve single-click segmentation. The user places one click on the object of interest, and the system

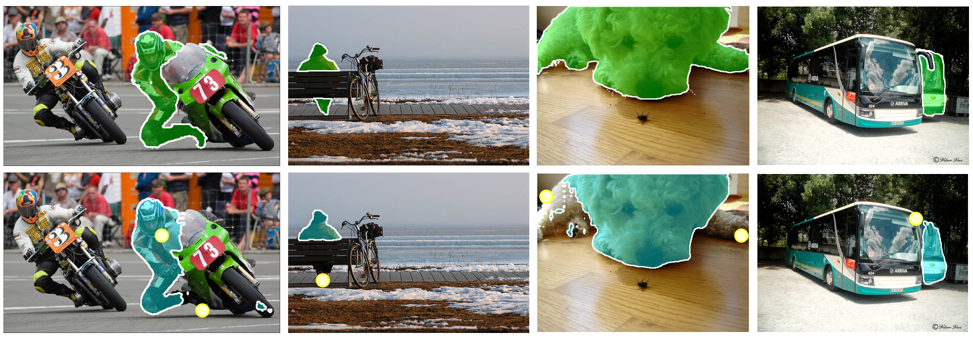


Figure 6.1: Two-in-One Refinement. Our two-in-one refinement strategy places refinement clicks on the target boundary of error regions. The top row shows sample images with ground truth masks in green. The second row shows an initial segmentation from the *selection* stage overlaid in cyan, with yellow circles marking refinement click locations.

returns an accurate mask. Despite significant progress, advanced deep learning-based frameworks (Liew et al., 2019; Sofiiuk et al., 2020; Kontogianni et al., 2020) still require multiple clicks: one to indicate the object of interest, and others to refine the segmentation mask. In all interactive frameworks (Xu et al., 2016; Mahadevan et al., 2018; Liew et al., 2017; Hu et al., 2019; Liew et al., 2017; Jang and Kim, 2019), refinement is done via a succession of positive and negative clicks. Intuitively, the positive clicks should add portions of missing foreground, while negative clicks remove falsely segmented parts of the background. Users place clicks centrally in regions that directly corresponds with false negatives and false positives. Distinguishing between the two forms of refinement, however, requires separate input encodings, so there are always at least two guidance maps (Xu et al., 2016; Liew et al., 2019; Li et al., 2018; Liew et al., 2017). Additionally, a plateauing occurs, as the average per instance mIoU tends to stagnate beyond 5-6 clicks (Xu et al., 2016; Mahadevan et al., 2018; Liew et al., 2017).

In this work, we do away with positive and negative clicks and propose a novel *two-in-one* refinement strategy. We ask users instead to place clicks on the instance *boundary* in the vicinity of errors (see Fig. 6.1). Boundary clicks have more utility, as they provide a much stronger cue than clicks placed centrally in regions of false positives and false negatives. Furthermore, as more and more boundary refinement clicks become available, the instance gets explicitly encircled with refinement clicks.

To accommodate our new refinement scheme, we use dedicated networks for object *selection* versus *refinement* (see Fig. 6.2). Most previous works have used a single network (Xu et al., 2016; Mahadevan et al., 2018; Liew et al., 2017; Jang and Kim, 2019; Sofiiuk et al., 2020), treating the initial selection click simply as any other positive click. In our case, the initial selection click is placed centrally within the object (Liew et al., 2019), while refinement clicks are placed on object boundaries. As a result, task separation becomes necessary. Having two networks does add some computation overhead in training and inference. To mitigate the impact, we share feature representations across the networks (see Fig. 6.2). The final refinement network is a lightweight block using three successive convolutions (see Sec. 6.3.2). In return, separating the tasks allows more freedom for the networks to specialize and thereby achieves state-of-the-art results.

For learning the selection network, we propose a new and highly effective boundary-aware cross-entropy loss. This loss encourages predicted segmentation masks to remain close to the instance boundaries, and through a *single* selection click, restricts the working image space. In contrast, previous works require two to four clicks. We find that isolating the region of interest for segmentation is particularly helpful for small objects, which previous works (Majumder and Yao, 2019a; Kontogianni et al., 2020) have already shown are the most difficult to segment.

The key contributions of our approach are summarized as follows:

- We propose a novel *two-in-one* refinement strategy for interactive segmentation where users refine segment masks by clicking on the instance boundary.
- We propose a boundary-aware cross-entropy loss which constrains the prediction and allows us to integrate a *crop* mechanism without the use of bounding boxes (Benenson et al., 2019) or explicit confining clicks at extremal points (Maninis et al., 2018; Benenson et al., 2019).
- A framework that separates selection and refinement tasks, with which we achieve relative mIoU gains of 10-15% over state-of-the-art approaches that use even deeper backbones.

6.2 Related Works

Deep-learning approaches (Mahadevan et al., 2018; Liew et al., 2019) based on fully convolutional network architectures (Long et al., 2015; Chen et al., 2018c,b) now excel at producing good quality segmentation masks even for challenging large-scale datasets (Everingham et al., 2010; Lin et al., 2014). With user-provided cues such as bounding boxes (Benenson et al., 2019), clicks (Xu et al., 2016; Mahadevan et al., 2018; Benard and Gygli, 2017), and scribbles (Agustsson et al., 2019), these approaches can generate instance segmentation masks with over 90% mean Intersection over Union (mIoU) w.r.t ground truth with less than four user clicks (Liew et al., 2019; Majumder and Yao, 2019a; Kontogianni et al., 2020).

Of these, click-based interactive frameworks (Xu et al., 2016; Mahadevan et al., 2018; Liew et al., 2017; Jang and Kim, 2019; Li et al., 2018) have particularly gained popularity in the last five years. The initial work of Xu et al. (2016) encoded user clicks as Euclidean distance maps; the maps are then concatenated with the RGB channels and fed as input to an FCN (Long et al., 2015). Based on the previous prediction errors, further clicks are added, usually on the center of the largest incorrect region (Benenson et al., 2019; Liew et al., 2019).

Across most of these approaches, user clicks are transformed guidance maps via Euclidean distance (Xu et al., 2016; Li et al., 2018) or Gaussians (Mahadevan et al., 2018). Alternatively, superpixels (Chen et al., 2018a; Majumder and Yao, 2019a,b) and region-based object proposals (Majumder and Yao, 2019a) have also been used to generate content-aware guidance maps. Given an initial bounding box, Polygon-RNN (Castrejon et al., 2017) predicts the vertices of the polygon outlining the object. Other considerations for interactive segmentation frameworks include a two-stream network (Hu et al., 2019) and backpropagation refinement scheme (Jang and Kim, 2019; Sofiiuk et al., 2020).

Historically, many classical approaches for interactive segmentation (Le et al., 2018; Mortensen and Barrett, 1995; Li et al., 2004; Kass et al., 1988; Gleicher, 1995) segmented objects by allowing users to interact with boundary pixels. Early variants used boundary tracing (Kass et al., 1988; Mortensen and Barrett, 1995; Falcao et al., 1998) but as such approaches relied solely on low-level image features such as gradients (Gleicher, 1995), they do not work well on unconstrained images. More recently, the DEXTR framework (Maninis et al., 2018) proposed using extremal boundary clicks to select and enclose the object. As DEXTR (Maninis et al., 2018) requires a minimum of four clicks at the outset, it requires significantly more user input than state-of-the-art methods (Liew et al., 2019; Majumder and Yao, 2019a; Jang and Kim, 2019; Kontogianni et al., 2020).

Recently, Le et al. (2018) proposed a deep learning-based framework for interactively finding object boundaries. Unlike Le et al. (2018) and DEXTR, we treat boundary clicks as a means of refining or correcting prediction errors. Click locations are neither arbitrary (Le et al., 2018) nor constrained to extremal points (Maninis et al., 2018) but conditioned by prediction errors of the preceding selection network (see Fig. 6.2). Additionally, we do not make use of traditional boundary prediction losses (Yu et al., 2017; Le et al., 2018). Finally, we report results on the more challenging Pascal VOC 2012 (Everingham et al., 2010) and MS COCO (Lin et al., 2014) datasets.

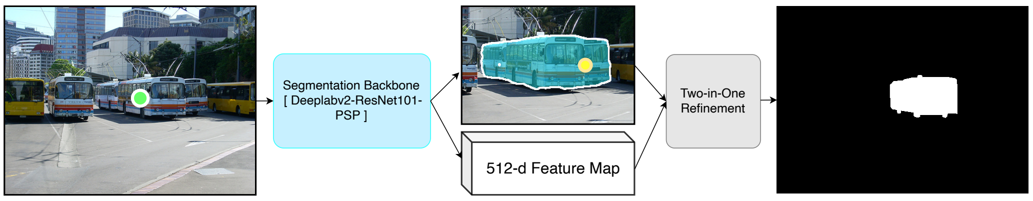


Figure 6.2: Outline. Given an initial selection click (shown in green) on the object of interest, our selection network generates a full-image segmentation, cropping coordinates and learned feature representations. Given the initial segmentation, the user refines with clicks on the boundary of the object in the cropped image and the segmentation mask is updated. Further refinement clicks (in yellow) can be added until a suitable mask is obtained.

6.3 Proposed Method

We follow the conventional paradigm of (Xu et al., 2016; Liew et al., 2017; Benard and Gygli, 2017; Majumder and Yao, 2019a; Jang and Kim, 2019) which transforms user clicks into *'guidance'* maps of the same size as the input image. The guidance maps are appended as extra channels to the RGB image and given as input to an FCN (Long et al., 2015). To generate the guidance maps, we use Gaussian transformations (Maninis et al., 2018; Benard and Gygli, 2017) as it offers a favourable trade-off between simplicity and performance. We initialize an image-sized channel of zeros and place Gaussians with a small standard deviation of 10 pixels at each user click location.

During training, we pass the RGB image channels concatenated with the guidance map from the first click through the selection network. The selection network returns the initial segmentation mask and corresponding feature maps. Based on the initial masks, we crop the prediction and the feature maps. We then determine refinement clicks from the crops, transform them into a guidance map, and append the guidance maps with the cropped feature maps. Finally, we process with a lightweight refinement network to generate a refined segmentation mask. Further refinement clicks can be added iteratively to reach an acceptable segmentation quality. Fig. 6.2 illustrates this process.

6.3.1 Selection Network

We initialize our selection network with the weights from Deeplabv2 (Chen et al., 2018b) pre-trained on Pascal VOC 2012 (Everingham et al., 2010) for semantic segmentation. Deeplabv2 uses a ResNet-101 (He et al., 2016) as the feature extraction backbone and PSP module (Zhao et al., 2017) for performing multi-scale feature aggregation. Extracted features from Deeplabv2 has 512 channels. The final 1-channel prediction logits come from a 1×1 convolution. We modify the final layer of Deeplabv2 (Chen et al., 2018b) to additionally return the 512-channel features, which we later reuse in the refinement network. We note that other backbones (Long et al., 2015; Chen et al., 2018c) can be used for the selection network as well.

Typically, interactive frameworks are trained to minimize the standard class-balanced binary cross-entropy loss is given by,

$$\mathcal{L}_{\text{CB-CE}} = \sum_{p \in P} w_{y_p} \cdot \text{BCE}(y_p, \hat{y}_p). \tag{6.1}$$

Here P is the number of pixels in the image, BCE$(\cdot)$ is the standard cross-entropy loss between the label y_p and the prediction $\hat{y}_p$ at pixel location p (Long et al., 2015). w_{y_p} denotes the inverse normalized frequency of ground truth labels $y_p \in \{0, 1\}$ of the mini-batch. We observe that for Eqn. 6.1, all pixels, irrespective of their location, contribute equally to the loss function. This often leads to noisy and non-continuous segment predictions (Audebert et al., 2019; Xue et al., 2020) that require further post-processing (Xu et al., 2016; Benard and Gygli, 2017). To circumvent this, several methods have explored learning from the distance transform (Ye, 1988; Xu et al., 2016) via additional regression losses (Xue et al., 2020; Audebert et al., 2019).

To this end, we propose a boundary-aware cross entropy loss (BA-CE) which selectively increases the penalty for incorrect classification of background pixels lying within some distance (in pixels) of the target instance. Unlike (Calivá et al., 2019), we opt for asymmetric penalty formulation. This focuses the network on learning instance-background transitions and prevents disjoint predictions far-off from the target instance. Formally, let B $(\in \mathcal{M})$ denote the set of boundary pixels for the ground truth mask $\mathcal{M}$. We define a per-pixel weighting factor $w_{(B,\,p)}$ as a function of the minimum Euclidean distance from the set of boundary pixels $p_b \in B$.

$$w_{(B,\,p)} = \begin{cases} 0 & d(B,\,p) > \tau \;\text{ or }\; p \in \mathcal{M} \\ \frac{d(B,\,p)}{k} & d(B,\,p) \leq \tau \;\text{ and }\; p \notin \mathcal{M} \end{cases} \tag{6.2}$$

where $d(B,\,p) = \min \|p - p_B\|_2^2 \;\; \forall\, p_B \in B$. τ denotes the width along the instance boundary (in pixels) where the penalty term is enforced. In our experiments, we fix $k = 255$ and $\tau = 40$. Our proposed BA-CE loss function is given as,

$$\mathcal{L}_{\text{BA-CE}} = \sum_{p \in P} w_{y_p} \cdot (1 + w_{(B,\,p)}) \cdot \text{BCE}(y_p, \hat{y}_p). \tag{6.3}$$

6.3.2 Refinement Network

Given the first selection click, the selection network returns 512-channel feature map $\mathcal{F} \in \mathbb{R}^{h\times w\times 512}$ and the predicted mask trimmed (see Fig. 6.2). Based on the trimmed predicted mask, the user annotates additional refinement clicks to undo the prediction errors. We encode these user-provided refinement clicks using Gaussians in a single channel $\mathcal{G} \in \mathbb{R}^{h\times w\times 1}$. We concatenate $\mathcal{F}$ and $\mathcal{G}$ along the feature-map dimension pass it through our lightweight refinement network.

For cropping the initial prediction from the selection network to the target instance, we make use of the RoIAlign layer (He et al., 2017) to obtain $\mathcal{F}_t$ from $\mathcal{F}$. RoIAlign (He et al., 2017) takes as input the region of interest (ROI) coordinates and the feature map and produces a fixed-size representation regardless of ROI size. Instead of RoIAlign (He et al., 2017), naive cropping is also an alternative. We obtain the cropping coordinates by first applying a sigmoid operator on the initial prediction logits and then selecting pixels with values higher than a specific threshold. We pass these coordinates to the RoIAlign layer and obtain features $\mathcal{F}_t$ for the cropped representation. We then concatenate $\mathcal{F}_t$ with $\mathcal{G}_{\mathcal{F}_t}$ and pass it through the refinement network. $\mathcal{G}_{\mathcal{F}_t}$ denotes the user click encoding on the cropped initial prediction. During training, we observed that naively processing raw features $\mathcal{F}_t$ through the refinement network is detrimental. Thus, we perform a channel-wise normalization of $\mathcal{F}_t$ before concatenating it with $\mathcal{G}_{\mathcal{F}_t}$.

Our refinement network consists of 3 convolutional blocks. We denote the i-th block as $C_i(m, n, k)$, which consists of a $k\times k\times m\times n$ convolution followed by batch normalization and ReLU. Here, k refers to kernel size, m the number of input feature channels and n the number of output feature channels. The architecture of our refinement layer is given by,

$$\Big[C_1(513, 512, 3) \rightarrow\ C_2(512, 256, 3) \rightarrow\ C_3(256, 256, 3)\Big] \tag{6.4}$$

The final prediction logits are obtained via 1×1 convolution which squashes the 256-channel into a single channel output. Threshold selection and instance recall are discussed in more detail in Sec. 6.4.2. We note that our selection network is not trained to minimize the bounding box regression loss (He et al., 2017) as per standard instance segmentation.

6.3.3 Click Sampling

Like previous works (Xu et al., 2016; Mahadevan et al., 2018; Chen et al., 2018a; Hu et al., 2019; Liew et al., 2019; Benard and Gygli, 2017), we simulate clicks for training and evaluation.

Selection click sampling. The initial *selection* click is typically assigned on the center of the target object (Liew et al., 2019). For convexly shaped instances, the center of mass computed using the ground truth serves as a prior for click initialization. In concave shapes, we shift the initialization to an interior point furthest from the instance boundary. To mimic humans in placing clicks and make our training robust, we further add random displacements of up to 25 pixels to the sampled click location. Visualizations of selection click placements are shown in Fig. 6.3.

Refinement click sampling. Refinement clicks are sampled according to segmentation outputs from the selection network. Given the prediction and ground truth masks, false positive and false negative regions are computed. The erroneous regions are then clustered based on connectivity (Mahadevan et al., 2018) while discarding regions with pixels fewer than $m\%$ of the ground truth. We then select a minimum of k_1 clicks and a maximum of k_2 from the remaining error regions. In our experiments, we vary m between 1-5 while k_1 and k_2 fixed at 2 and 4 respectively i.e., the network sees 2-4 refinement clicks to refine each instance.

For sampling refinement clicks corresponding to the false negative regions, a one-sided (directed) Hausdorff distance is used. Let $\mathcal{B}_{FN}$ and π_{FN} denote the set of ground truth boundary pixels and the predicted boundary pixels outlining the false negative region respectively. For the false positive clusters, we opt for the $\min - \min$ formulation. Likewise, let $\mathcal{B}_{FP}$ and π_{FN} denote the set of ground truth boundary pixels and the predicted boundary pixels outlining the corresponding false positive region respectively. The sampled click locations $c_{b,\,FN} \in \mathcal{B}_{FN}$ and $c_{b,\,FP} \in \mathcal{B}_{FP}$ corresponding to false negative and false positive regions are given respectively by,

$$c_{b,FN} = \arg\max_{x \in \mathcal{B}_{FN}} \min_{y \in \pi_{FN}} \|x - y\|_2 \qquad \text{and} \qquad c_{b,FP} = \arg\min_{x \in \mathcal{B}_{FP}} \min_{y \in \pi_{FP}} \|x - y\|_2. \quad (6.5)$$

During inference, it is unlikely that users clicks will align exactly with the object boundary. During training, we randomly perturb the click location up to 10 pixels to compensate.

6.3.4 Implementation Details

Similar to (Mahadevan et al., 2018; Xu et al., 2016; Kontogianni et al., 2020; Majumder and Yao, 2019a; Liew et al., 2017), we simulate user click behavior and use the 1464 training images from Pascal VOC 2012 (Everingham et al., 2010) plus the additional instance annotations from SBD provided by (Hariharan et al., 2011). We further augment with random scaling, flipping, and rotation operations. Unlike (Liew et al., 2019; Mahadevan et al., 2018), we do not use training instances from MS COCO (Lin et al., 2014).

For training both the selection and refinement network, we minimize the BA-CE loss (Eqn. 6.3) using stochastic gradient descent with Nesterov momentum (fixed at the default value of 0.9) and apply a fixed learning rate of 10^{-8} across all epochs with weight decay of 0.0005. Training of the selection network typically converges within 80 epochs. For training the refinement network, we pick training instances with 0.2-0.6 mIoU based on predictions by the selection network. Training requires 10-15 epochs; to make the network more responsive to refinement clicks, we increase the pixel contribution to the loss by a scalar factor of 2 within a $40{\times}40$ neighbourhood of the sampled refinement click location.

6.4 Experiments

6.4.1 Datasets & Evaluation

We evaluate our proposed approach on five publicly available datasets. **Grabcut** (Rother et al., 2004) and **Berkeley** (McGuinness and O'connor, 2010) are have 49 and 96 images respectively, and feature mostly a single prominent foreground object. **Pascal VOC 2012** (Everingham et al., 2010) has 3427 instances across 1449 validation images spread across 20 object categories. **MS COCO** has 5000 validation (val2017) images over 80 categories, 20 of which overlap with Pascal. Like (Xu et al., 2016; Liew et al., 2017; Kontogianni et al., 2020), we pick 10 images randomly per category for evaluation and tabulate the 20 seen Pascal versus 60 other unseen categories separately. The **DAVIS 2016** dataset (Perazzi et al., 2016) consists of 50 high-resolution videos; following (Li et al., 2018; Jang and Kim, 2019), we randomly sample 10% of the frames to get 345 images.

In existing literature, non-interactive semantic segmentation (Chen et al., 2018b,c; Long et al., 2015) frameworks are evaluated with mean intersection over union (mIoU) (Everingham et al., 2010) by comparing the ground truth mask with the segmentation predictions of the network. Unlike semantic segmentation, in interactive segmentation, user interactions aimed to improve the current network prediction are available. Typically, such interactions come in the form of clicks and should increase the mIoU of the predicted mask by fixing errors in foreground or background areas.

We evaluate our framework with the two standard metrics (Benard and Gygli, 2017; Mahadevan et al., 2018; Xu et al., 2016) (a) **clicks@mIoU%** which is the average number of clicks needed to reach fixed mIoU on each instance and (b) **mIoU%@clicks**, the mean intersection over union given k user-assigned selection and/or refinement clicks per instance. In keeping with existing approaches (Benard and Gygli, 2017; Liew et al., 2017; Mahadevan et al., 2018; Xu et al., 2016; Liew et al., 2019; Majumder and Yao, 2019a), we threshold the number of clicks per

Dataset	ITIS	*with* CB-CE	*with* BA-CE
GrabCut (Rother et al., 2004)	82.1	84.0	**84.8**
Pascal VOC 2012 (Everingham et al., 2010)	71.0	78.2	**80.6**

Table 6.1: Selection Network. Average mIoU over all instances in the GrabCut and Pascal VOC 2012 *val* set obtained using ITIS (Mahadevan et al., 2018) and our frameworks; one network trained using a standard class-balanced cross-entropy loss (CB-CE) and another using our newly proposed boundary-aware cross-entropy (BA-CE) given only one (selection) click. Best results in **bold**.

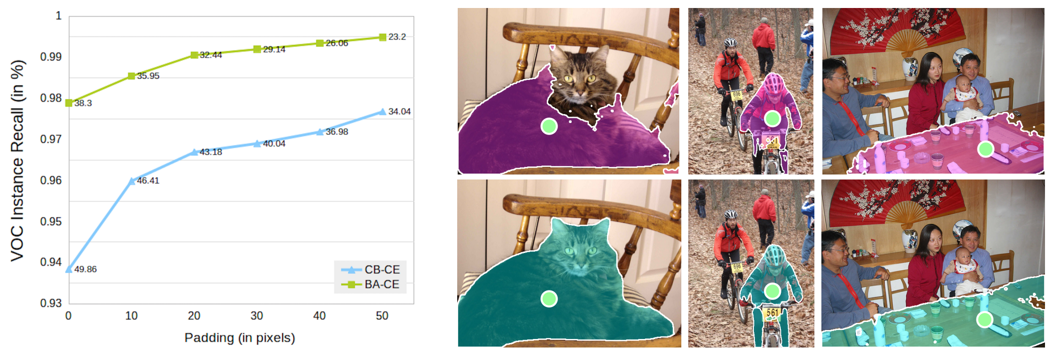
instance to 20. The minimum mIoU threshold is set at 90% for the GrabCut and Berkeley dataset. For the more challenging DAVIS, Pascal and MS COCO datasets, the threshold is 85% mIoU (Xu et al., 2016; Liew et al., 2017; Benard and Gygli, 2017; Majumder and Yao, 2019a; Hu et al., 2019).

6.4.2 Selection Network

We train two selection networks to minimize the standard class-balanced BCE (Eqn. 6.1) and our proposed variant (Eqn. 6.3). We observe an outright gain of 2.4% over the Pascal VOC 2012 *val* set (Everingham et al., 2010) and significant mIoU gains over recent state-of-the-art methods (Mahadevan et al., 2018) across the three datasets for the network trained with the BA-CE loss (Eqn. 6.3). The most significant aspect of the BA-CE loss function is its ability to constrain the network prediction; this is a desirable and necessary property which allows us to crop to the target instance. With a single click, we achieve 80.6% mIoU averaged over Pascal VOC 2012 (see Table. 6.1).

For further evaluation, we apply the sigmoid operation on the 1-channel logits predicted by both variants of the selection networks and crop the image region to pixel values > 0.001. To avoid tight cropping, which can lead to undesired boundary artifacts, we relax the cropped region via padding. The instance recall rate for all the validation instances in the Pascal (Everingham et al., 2010) dataset *vs* padding in pixels is shown in Fig. 6.3(a). The BA-CE network successfully recalls 99.6% instances, i.e. the cropped feature map retains the entire object 3414 times out of 3427; for the other 13 instances, we assume a penalty of 20 clicks.

We additionally plot the area reduction (in %-age of the image area) after cropping given a fixed padding (shown in Fig. 6.3(a)). We observe that the network trained to minimize the CB-CE loss (Eqn. 6.1) offers more area reduction at the cost of a lower recall. However, this network is undesirable, as we intend to segment all annotated instances in the Pascal VOC 2012 *val* set (Everingham et al., 2010).



(a) VOC Instance Recall in % *vs* padding

(b) Prediction results with CB-CE (Row 1) *vs* BA-CE (Row 2)

Figure 6.3: CB-CE *vs* BA-CE. Comparison of the class balanced cross-entropy (CB-CE) (Maninis et al., 2018) with our proposed boundary aware cross-entropy (BA-CE) loss. Green points denote initial selection clicks.

Qualitative results are shown in Fig. 6.3(b). We observe that BA-CE encourages spatially coherent segmentation masks.

6.4.3 Comparison to the state-of-the-art

We compare the **clicks@mIoU%** performance of our method against competing methods (Xu et al., 2016; Liew et al., 2017, 2019; Majumder and Yao, 2019b; Jang and Kim, 2019) in see Table. 6.2. As an ablation study to verify the effectiveness of task separation in the interactive workflow, we also test a variant where the refinement network is trained to accept iterative positive and negative clicks, as per (Xu et al., 2016; Mahadevan et al., 2018; Liew et al., 2017; Majumder and Yao, 2019b,a; Liew et al., 2019) (Ours-*PNR*). In Ours-*PNR*, based on an initial mask from the selection network, users iteratively provide positive and negative clicks; the clicks are then encoded as Gaussians in two separate channels. The rest of the pipeline is unchanged.

This variant outperforms several competing methods, suggesting that existing frameworks can also benefit by task splitting across dedicated networks. Our full variant using the two-in-one refinement (Ours-*BRC*) achieves state-of-the-art on almost all the benchmarks. Our boundary-based refinement offers a 10% improvement in comparison to Ours-*PNR*.

We significantly raise the bar on the Berkeley (McGuinness and O'connor, 2010), Pascal VOC 2012 *val* set (Everingham et al., 2010), and DAVIS 2016 (Perazzi et al., 2016) datasets by 6.5%, 9.4% and 10.5% respectively. Additionally, we outperform current state-of-the-art (Majumder and Yao, 2019b) on MS COCO (Lin et al., 2014) *seen* and *unseen* by around 2.5% and 1% respectively. Additionally, we are

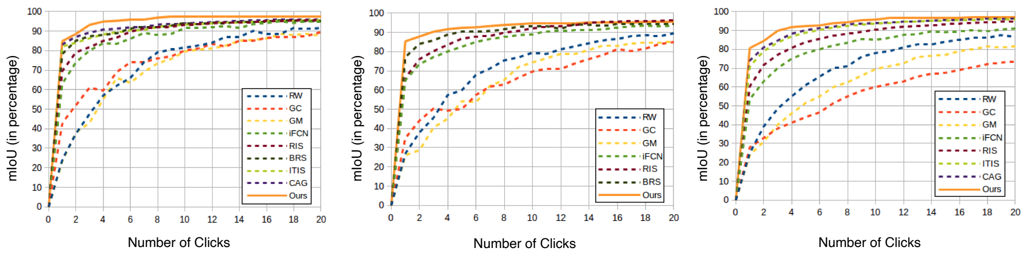
Method	Grabcut @90%	Berkeley @90%	VOC-2012 @85%	DAVIS @85%	COCO-20 @85%	COCO-60 @85%
iFCN$_{\text{CVPR'16}}$	6.04	8.65	6.88	-	8.31	7.82
RIS$_{\text{ICCV'17}}$	5.00	6.03	5.12	-	5.98	6.44
ITIS$_{\text{BMVC'18}}$	5.60	-	3.80	-	-	-
CAG$_{\text{CVPR'19}}$	3.58	5.60	3.62	-	5.40	6.10
LIS$_{\text{GCPR'19}}$	3.46	5.18	3.70	-	5.15	5.70
BRS$_{\text{CVPR'19}}$	3.60	5.08	-	5.58	-	-
MSG$_{\text{ICCV'19}}$	1.96	4.00	3.51	-	-	-
LFC$_{\text{ECCV'20}}$	3.07	4.94	3.18	5.57	9.14	9.87
Ours-*BCR*	2.30	**3.74**	**2.88**	**4.98**	**5.02**	**5.65**

Table 6.2: Average number of clicks required to achieve a fixed mIoU across state-of-the-art deep learning-based interactive frameworks - iFCN (Xu et al., 2016), RIS (Liew et al., 2017), ITIS (Mahadevan et al., 2018), CAG (Majumder and Yao, 2019a), LIS (Majumder and Yao, 2019b), BRS (Jang and Kim, 2019), LFC (Kontogianni et al., 2020), MSG (Liew et al., 2019). Best results are indicated in **bold**. COCO-20 and COCO-60 refers to the instances from 20 overlapping and 60 non-overlapping *w.r.t* Pascal VOC 2012 object categories respectively.

comparable with current state-of-the-art MSG (Liew et al., 2019) on the GrabCut dataset (Rother et al., 2004). In Fig. 6.4, we plot our average mIoU *vs* the number of clicks against competing methods RW (Grady et al., 2005), GC (Boykov and Jolly, 2001), GM (Bai and Sapiro, 2009), iFCN (Xu et al., 2016), RIS (Liew et al., 2017), ITIS (Mahadevan et al., 2018), CAG (Majumder and Yao, 2019a). Task-separation give us a head-start *w.r.t* all existing approaches for the first click placement. Overall, our approach shows better performance in most cases.

6.4.4 User study

We validate the use of boundary clicks for refinement with a user study, using 50 images from Pascal VOC *val* set with mIoU between 60%-75% as predicted by the selection network. We overlay the prediction of the selection network and also presented the ground truth mask as a separate image. Our 5 participants are tasked with providing refinement clicks on the boundary pixels corresponding to the largest error regions. On average, participants correctly identified the largest erroneous region with a mean accuracy of 88%. Errors may arise when multiple blobs have similar sizes; for example, in Fig. 6.2, the erroneous region on the left and right boundary of the bus is almost similar in size. Click placements were typically fast, with a mean ± standard deviation of 3.38±0.8 seconds. Click placements were also quite accurate and fell within 2-5 pixels of the instance boundary. For comparison,



(a) GrabCut (b) Berkeley (c) Pascal VOC 2012

Figure 6.4: mIoU%@clicks across 3 datasets (GrabCut, Berkeley, Pascal VOC 2012) for our approach *vs* 8 competing interactive methods - RW (Grady et al., 2005), GC (Boykov and Jolly, 2001), GM (Bai and Sapiro, 2009), iFCN (Xu et al., 2016), RIS (Liew et al., 2017), ITIS (Mahadevan et al., 2018), CAG (Majumder and Yao, 2019a). RW, GC, and GM are *classical* interactive methods, whereas iFCN, RIS, ITIS, and CAG are FCN-based interactive frameworks.

for each instance, DEXTR (Maninis et al., 2018) acquired four extremal clicks for each object instance within 7 seconds.

6.5 Conclusion

In this chapter, we have proposed a new two-in-one refinement strategy for interactive object segmentation. Instead of conventional positive and negative clicks to fix false negative and false positive segmentations, we ask users to click on object boundaries. To accompany this new refinement scheme, we propose a new boundary-aware loss formulation and a task-specialized framework where dedicated networks are applied to select and refine the object. The refinement network is lightweight and reuses feature maps from the selection network, so additional computational overhead minimal. We find that boundary clicks coupled with our boundary-aware loss provide a strong cue for interactive segmentation and raises state-of-the-art on several benchmark datasets.

CHAPTER 7

Multi-Stage Fusion for One-Click Segmentation

This chapter includes the content of the following publication:

- Soumajit Majumder, Anshul Khurana, Abhinav Rai, and Angela Yao. "Multi-Stage Fusion for One-Click Segmentation." *DAGM German Conference on Pattern Recognition (GCPR)*, 2020.

Segmenting objects of interest in an image is an essential building block of applications such as photo-editing and image analysis. Under interactive settings, one should achieve good segmentations while minimizing user input. Current deep learning-based interactive segmentation approaches use early fusion and incorporate user cues at the image input layer. Since segmentation CNNs have many layers, early fusion may weaken the influence of user interactions on the final prediction results. In this chapter, we propose a new multi-stage guidance framework for interactive segmentation. By incorporating user cues at different stages of the network, we allow user interactions to impact the final segmentation output in a more direct way. Our proposed framework has a negligible increase in parameter count compared to early-fusion frameworks. We perform extensive experimentation on the standard interactive instance segmentation and one-click segmentation benchmarks and report state-of-the-art performance.

7.1 Introduction

The widespread availability of smartphones had made taking photos easier than ever. In a typical image capturing scenario, the user taps the device touchscreen to focus on the object of interest. This tap directly locates the object in the scene and can be leveraged for segmentation. Generated segmentations are implicit, but are applicable for downstream photo applications, such as simulated 'bokeh' or other special-effects filters such as background blur (see Fig. 7.1). In this work, we specifically tackle *"tap-and-shoot segmentation"* (Chen et al., 2018a), a special case of interactive instance segmentation.

Interactive segmentation leverages inputs such as clicks, scribbles, or bounding boxes to help segment objects from the background down to the pixel level. Two

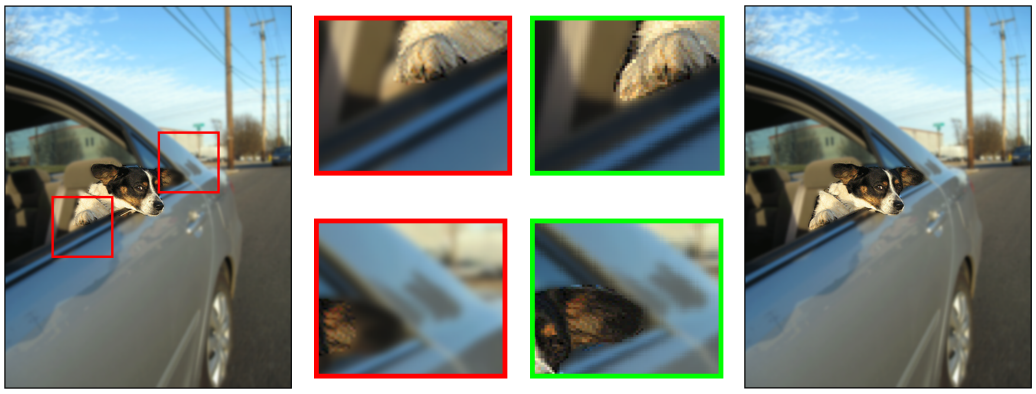


Figure 7.1: We consider the popular special-effect filter used in mobile photography - background blur. Here the user intends to blur the rest of the image barring the dog. In most existing interactive segmentation approaches (Xu et al., 2016; Liew et al., 2017; Mahadevan et al., 2018; Sofiiuk et al., 2020), the user click (here placed on the dog) is leveraged only at the input layer and its influence diminishes through the layers. This can result in unsatisfactory image effects, e.g portions of the dog's elbow and ear are wrongly classified as background and are mistakenly blurred (shown in enlarged red boxes). Our proposed multi-stage fusion allows user click to have a more direct effect leading to improvement in segmentation quality (shown in enlarged green boxes).

key differences distinguish tap-and-shoot segmentation from standard interactive segmentation. *First*, tap-and-shoot uses only *"positive"* clicks marking foreground, as we assume that the user clicks (only) on the object of interest during the capture process. Standard interactive segmentation uses both positive and negative clicks (Xu et al., 2016; Mahadevan et al., 2018; Sofiiuk et al., 2020) to respectively indicate the object of interest versus background or non-relevant foreground objects. *Secondly*, tap-and-shoot has a strong focus on maximizing the mean intersection over union (mIoU) with a single click because the target application is casual photography. In contrast, standard interactive segmentation tries to achieve some threshold mIoU (e.g. 85%) while minimizing the total number of clicks.

This second distinction is subtle but critical for designing and learning tap-and-shoot segmentation frameworks. Our finding is that existing approaches fare poorly with only one or two clicks – they are simply not trained to *maximize* performance under such settings. To make the most of the first (few) click(s), we hypothesize that user cues' guidance should be fused into the network at multiple locations rather than via early fusion. Just as gradients vanish towards the initial layers during back-propagation, input signals also diminish as it makes a forward pass through the network. The many layers of deep CNNs further exacerbate this effect (Hu et al., 2019; Park et al., 2019). A late fusion would allow the user interaction to have a direct and more pronounced effect on the final segmentation mask. To this end, we propose an interactive segmentation framework with multi-stage fusion and demonstrate its advantages over the common early fusion frameworks and other alternatives. Specifically, we propose a light-weight fusion block that encodes the user click transformation and allows a shorter connection from user inputs to the final segmentation layer.

Most similar in spirit to our framework are the frameworks of Hu et al. (2019) and Rakelly et al. (2018). These two works also propose alternatives to early fusion but are extremely parameter heavy. For example, the framework of Hu et al. (2019) uses two dedicated VGG (Simonyan and Zisserman, 2014) networks to to extract features from the image and the user interactions separately before fusing into a final instance segmentation mask (see Fig. 7.2(c)). Rakelly et al. (2018) uses a single stream but applies a simple late fusion of element-wise multiplication on the feature maps (see Fig. 7.2(b)). It therefore has separate 'positive' and 'negative' feature maps and the number of weights for the following layer increases by a factor of 2. For VGG, this doubles the parameters of the ensuing *'fc6'* layer from 100 to 200 million. Compared to Rakelly et al. (2018), our last-stage fusion approach is light-weight and uses less than 1.5% more trainable parameters.

Our contributions are summarized as follows:

- We propose a novel one-click interactive segmentation framework that fuses user guidance at different network stages.
- We demonstrate that multi-stage fusion is highly beneficial for propagating guidance and increasing the mIoU since it allows user interaction to have a more direct impact on the final segmentation.
- Comprehensive experiments on six benchmarks show that our approach significantly outperforms existing state-of-the-art for both tap-and-shoot and standard interactive instance segmentation.

7.2 Related Works

As an essential building block of image/video editing applications, interactive segmentation and dates back decades (Mortensen and Barrett, 1995). The latest methods (Mahadevan et al., 2018; Xu et al., 2016; Hu et al., 2019; Rakelly et al., 2018) integrate deep architectures such as FCN-8s (Long et al., 2015) or Deeplab (Chen et al., 2018c,b). Most of these approaches integrate user cues in the input stage. The clicks are transformed into 'guidance' maps and appended to the three-channel colour image input before being passed through a CNN (Mahadevan et al., 2018; Xu et al., 2016; Majumder and Yao, 2019a).

Early **Interactive Instance Segmentation** approaches used graph-cuts (Rother et al., 2004), geodesics, or a combination (Gulshan et al., 2010). These methods' performance is limited as they separate the foreground and background based on low-level colour and texture features. Consequently, for scenes where foreground and background are similar in appearance, or lighting and contrast is low, more labelling effort from the users to achieve good segmentations (Xu et al., 2016). Recently, deep convolutional neural networks (Long et al., 2015; Chen et al., 2018b,c) have been incorporated into interactive segmentation frameworks. Initially, Xu et al. (2016) used Euclidean distance-based guidance maps to represent user-provided clicks and are passed along with the input RGB image through a fully convolutional network.

Subsequent works made extensions with newer CNN architectures (Mahadevan et al., 2018), iterative training procedures (Mahadevan et al., 2018) and content-aware guidance maps (Majumder and Yao, 2019a,b). These works share a structural similarity: the guidance maps are concatenated with the RGB image as additional channels at the first (input) layer. We refer to this form of structure as early fusion (see Fig. 7.2(a)). Architecture-wise, early fusion is simple and easy to train; however, user inputs' influence gets diminished through the layers.

Tap-and-Shoot Segmentation was introduced by (Chen et al., 2018a), and refers to the one-click interactive setting. One assumes that during image capture, the user taps the touchscreen (once) on the foreground object of interest, from which one can directly segment the object of interest. Chen et al. (2018a) uses early fusion; it transforms the user tap into a guidance map via two shortest-path minimizations and then concatenates the map to the input image. The authors validate on simple datasets such as ECSSD (Shi et al., 2015) and MSRA10K (Cheng et al., 2014), where the images contain a single dominant foreground object. As we show later (see Table. 7.1), these datasets are so simplistic that properly trained networks with *no* user input can also generate high-quality segmentation masks which are comparable or even surpass the results reported by (Chen et al., 2018a).

Feature Fusion in Deep Architectures offers an efficient way to leverage complementary information, either from different modalities (Vielzeuf et al., 2017), or different levels of abstraction (Zhang et al., 2019). Element-wise multiplication (Rakelly et al., 2018) and addition (Hu et al., 2019; Liu et al., 2019) are two common operations applied for fusing multiple channels. Other strategies include 'skip' connections (Long et al., 2015), where features from earlier layers

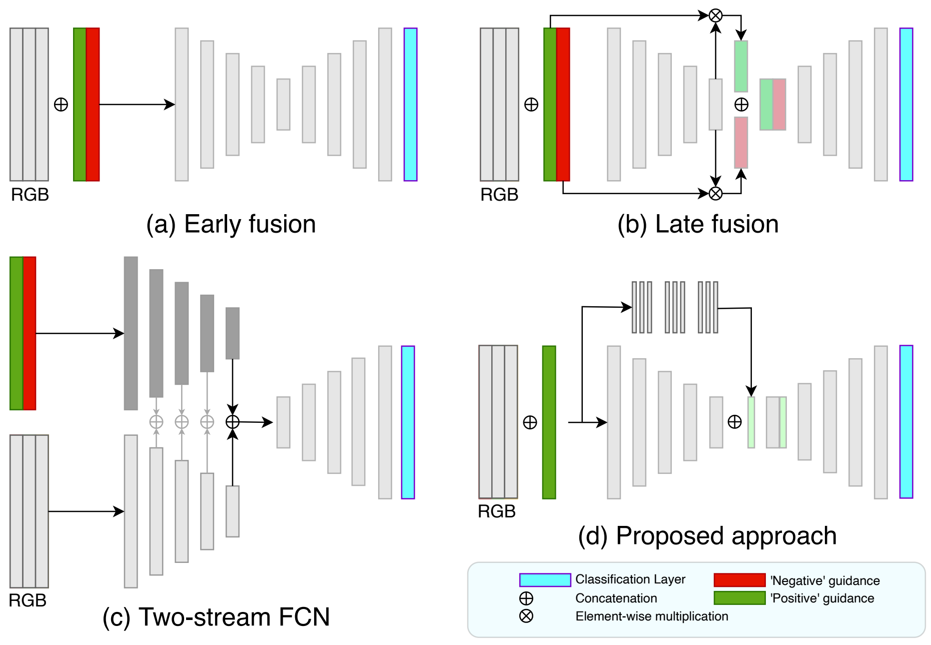


Figure 7.2: (a) Existing interactive instance segmentation and "tap-and-shoot" segmentation techniques concatenate user provided cues as an extra guidance map(s) (for 'positive' and 'negative' clicks) with the RGB and pass everything through a segmentation network. (b-c) Other alternative approaches are extremely parameter heavy. (b) The work of (Hu et al., 2019) uses two dedicated VGG (Simonyan and Zisserman, 2014) networks for extracting features from image and user interactions *separately*. (c) The work of (Rakelly et al., 2018) performs late fusion via element-wise multiplication on the feature maps which requires an additional 100 million parameters. (d) We leverage user guidance at the input (early fusion) and via late fusion. Our multi-stage fusion reduces the layers of abstraction and allows user interactions to have a more direct impact on the final output.

are concatenated with the features extracted from the deeper layers. Recently, a few interactive instance segmentation works have begun exploring outside of the early-fusion paradigm to integrate user guidance (Hu et al., 2019; Rakelly et al., 2018). However, these approaches are heavy in their computational footprint, as they increase the number of parameters to be learned by order of hundred of millions (Rakelly et al., 2018). Dilution of input information is common-place in deep CNNs as the input gets processed several blocks of convolution (Park et al., 2019). Feature fusion helps preserve input information by reducing the layers of abstraction between the user interaction and the segmentation output.

7.3 Proposed Method

7.3.1 Overview

We follow the conventional paradigm of (Xu et al., 2016) in which 'positive' and 'negative' user clicks are transformed into *'guidance'* maps of the same size as the input image. Unlike majority of the interactive frameworks (Xu et al., 2016; Liew et al., 2017; Mahadevan et al., 2018; Li et al., 2018; Sofiiuk et al., 2020), we work within the one-click setting. The user provides a single 'positive' click on the object of interest; this click is then encoded into a single channel guidance map $\mathcal{G}$. We then feed the 3-channel RGB image input and the guidance map as an additional channel into a fully convolutional network. Fig. 7.3(a) shows an overview of our pipeline. Typically these FCNs are fine-tuned versions of semantic segmentation networks such as FCN-8s (Long et al., 2015) or Deeplab (Chen et al., 2018b).

For our base segmentation network, we use Deeplabv2 (Chen et al., 2018b); it consists of a ResNet-101 (He et al., 2016) feature extraction backbone and a Pyramid Scene Parsing (PSP) module (Zhao et al., 2017) acting as the prediction head. Upon receiving the input of size $h \times w \times 4$, the ResNet-101 backbone generates feature maps with a stride of 8, i.e., of dimension $\frac{h}{8} \times \frac{w}{8} \times 2048$ (Fig. 7.3(a)).

7.3.2 Multi-stage fusion

The fusion module consists of 3 *Squeeze-and-Excitation* residual blocks (SE-ResNet) (Hu et al., 2018). Proposed in (Hu et al., 2018), SE-ResNet blocks have been shown to effective for a variety of vision tasks such as image classification on ImageNet (Russakovsky et al., 2015) and object detection on MS COCO (Lin et al., 2014). SE-ResNet blocks incur minimal additional computational overhead as they consist of two 3×3 convolutional layers, two inexpensive fully connected layers and channel-wise scaling operation. Each SE-ResNet block consists of a *residual* block, a *squeeze* operation which produces a channel descriptor by aggregating feature maps across their spatial operation, dimensionality reduction layer (by reduction ratio r) and an *excitation* operation which captures the channel interdependencies. The individual components of the SE-ResNet block is shown in Fig. 7.3(b). The residual block consists of two 3×3 convolutions, batch normalization, and a ReLU non-linearity (Fig. 7.3(c)). We fix the number of filter banks to be 256 for each of the 3×3 convolution. The reduction ratio r is kept as 16 (Hu et al., 2018). The input to the fusion block is a $h/4 \times w/4 \times 256$ feature map which is obtained by processing the $h \times w \times 4$ input with 7×7 convolution operation with stride 2, batch normalization, ReLU non-linearity (Maas et al., 2013) and a 2×2 max-pooling operation with a stride of 2 (*Init* block, Fig. 7.3(a)). The final SE-ResNet block downsamples to generate a $h/8 \times w/8 \times 256$ feature map. This is concatenated with the $h/8 \times w/8 \times 2048$ obtained from the feature extraction backbone to obtain a

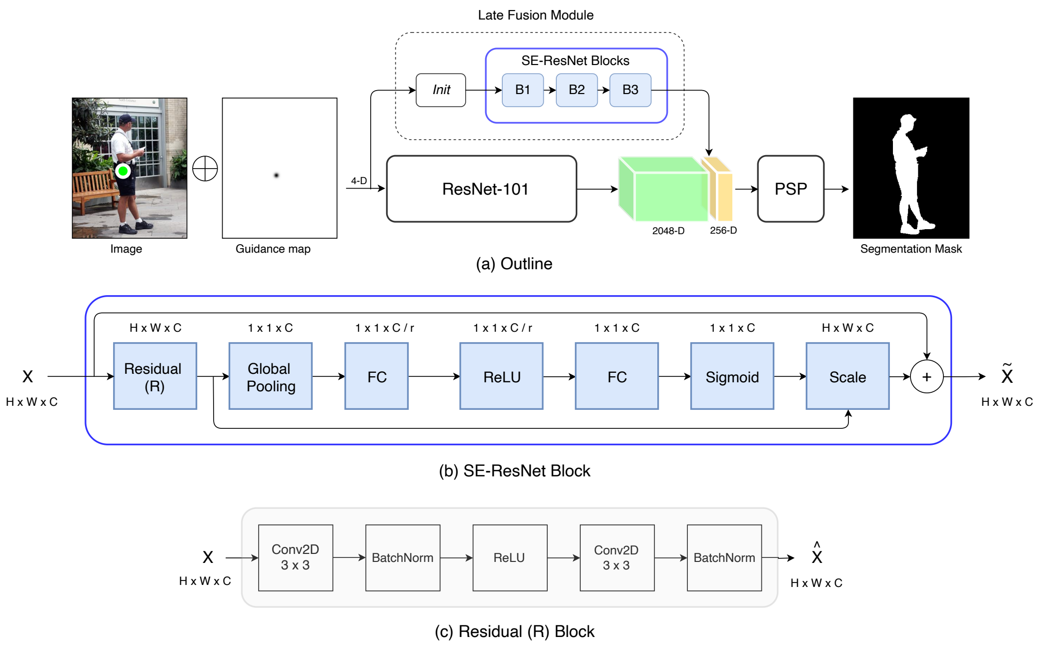


Figure 7.3: (a) Overview of our pipeline. Given an image and a 'positive' user click (shown in green circle), we transform the click into a Gaussian guidance map, which is concatenated with the 3-channel image input and is fed to our segmentation network. For ease of visualization, inverted values for the Gaussian guidance map is shown in the image. The output is the segmentation mask of the selected object. (b) SE-ResNet block (Hu et al., 2018) (c) Residual block in SE-ResNet.

$h/8 \times w/8 \times 2304$ feature map.

On top of these feature maps, PSP performs pooling operations at different grid scales on the feature maps to gather the global contextual prior, leading to feature maps of dimensions $h/8 \times w/8 \times 512$. The multi-scale feature pooling of PSP (Zhao et al., 2017) enables the network to capture objects occurring at different image scales. Pixel-wise foreground-background classification is performed on these down-sampled feature maps. The network outputs a probability map representing whether a pixel belongs to the object of interest or not. Bi-linear interpolation is performed to up-sample the predicted probability map to have the same dimensions as the original input image.

7.3.3 Transforming user click

In interactive approaches, pixel values of the guidance map are defined as a function of its distance on the image grid to the point of user interaction (Eqn. 7.1). This includes Euclidean (Xu et al., 2016; Hu et al., 2019) and Gaussian guidance maps (Mahadevan et al., 2018). For each pixel position $\boldsymbol{p}$ on the image grid, the pair of distance-based guidance maps for positive (+) and negative clicks (−) can be computed as

$$\mathcal{G}^d_+(\boldsymbol{p}) = \min_{\boldsymbol{c}\in\{\boldsymbol{p}_+\}} d(\boldsymbol{p},\boldsymbol{c}) \text{ and } \mathcal{G}^d_-(\boldsymbol{p}) = \min_{\boldsymbol{c}\in\{\boldsymbol{p}_-\}} d(\boldsymbol{p},\boldsymbol{c}). \tag{7.1}$$

For Euclidean guidance maps (Xu et al., 2016), the function $d(\cdot,\cdot)$ is the Euclidean distance. For Gaussian guidance maps, the 'min' is replaced by a 'max' operator. A more recent approach advocated taking image structures such as super-pixels and region-based object proposals into consideration to generate guidance maps (Majumder and Yao, 2019a,b). To generate the guidance maps, we use Gaussian transformations (Mahadevan et al., 2018) as it offers a favourable trade-off between simplicity and performance. We initialize an image-sized all zero channel and place a Gaussian with a standard deviation of 10 pixels at the user click location. Note that we do not use 'negative' clicks in our framework.

7.3.4 Implementation Details

Network Optimization. We train the network to minimize the class-balanced binary cross-entropy loss,

$$\mathcal{L} = \sum_{j\in N} w_{y_j} \cdot \mathrm{BCE}(y_j, \hat{y}_j) \tag{7.2}$$

where N is the number of pixels in the image, BCE$(\cdot)$ is the standard cross-entropy loss between the label y_j and the prediction $\hat{y}_j$ at pixel location j given by,

$$\mathrm{BCE}(y_j, \hat{y}_j) = -y_j \cdot \log\hat{y}_j - (1-y_j)\cdot\log(1-\hat{y}_j) \tag{7.3}$$

w_{y_j} is the inverse normalized frequency of labels $y_j \in \{0,1\}$ within the mini-batch. We optimize using mini-batch SGD with Nesterov momentum (with default value of 0.9) and a batch size of 5. The learning rate is fixed at 10^{-8} across all epochs and weight decay is 0.0005. For the ResNet-101 backbone, we initialize the network weights from a model pre-trained on ImageNet (Russakovsky et al., 2015). During training, we first update the early-fusion skeleton for 30-35 epochs. Next we freeze the weights of the early-fusion model and train the late-fusion weights for 5-10 epochs. Finally, we train the joint network for another 5 epochs.

Simulating user clicks. Manually collecting user interactions is an expensive and arduous process (Benenson et al., 2019). In a similar vein as (Chen et al., 2018a) and other interactive segmentation frameworks (Xu et al., 2016; Liew et al., 2017; Mahadevan et al., 2018; Jang and Kim, 2019; Sofiiuk et al., 2020), we simulate user interactions to train and evaluate our method. During training, we use the ground truth masks of the object instances from the MSRA10K dataset. To initialize, we take the center of mass of the ground truth mask as our user click location; we then jitter the click location by $\mathcal{U}(-50, 50)$ pixels randomly ($\mathcal{U}$ is the uniform distribution). The clicked pixel location is constrained to the confines of the object ground truth mask. The random perturbation introduces variation in the training data and also allows better approximation of true user interactions.

7.4 Experimental Validation

7.4.1 Datasets

We evaluate on six publicly available datasets commonly used to benchmark interactive image segmentation (Xu et al., 2016; Mahadevan et al., 2018; Chen et al., 2018a; Majumder and Yao, 2019a): MSRA10K (Cheng et al., 2014), ECSSD (Shi et al., 2015), GrabCut (Rother et al., 2004), Berkeley (McGuinness and O'connor, 2010), Pascal VOC 2012 (Everingham et al., 2010) and MS COCO (Lin et al., 2014). We use mean intersection over union (mIoU) of foreground w.r.t. to the ground truth object mask across all instances to evaluate the segmentation accuracy as per existing works (Long et al., 2015; Xu et al., 2016; Mahadevan et al., 2018; Chen et al., 2018a; Majumder and Yao, 2019a).

MSRA10K has 10000 natural images; the images are characterized by variety in the foreground objects whilst the background is relatively homogeneous. Extended complex scene saliency dataset (**ECSSD**) is a dataset of 1000 natural images with structurally complex backgrounds. **GrabCut** is a dataset consisting of 49 images with typically a distinct foreground object. It is a popular dataset for benchmarking interactive instance segmentation algorithms. The **Berkeley** dataset consists of 96 natural images. **Pascal VOC 2012** consists of 1464 training and 1449 validation images across 20 different object classes; many images contain multiple objects. **MS COCO** is a challenging large-scale image segmentation dataset with images from 80 different object categories, 20 of which are in common with the Pascal VOC 2012 categories.

7.4.2 Tap-and-Shoot Segmentation

Following Chen et al. (2018a), we use MSRA10K (Cheng et al., 2014) for training. We partition the dataset into three non-overlapping subsets of 8000, 1000 and 1000 images as our training, validation, and test set. We report the mIoU after training for 16k iterations and again after network convergence (at 43k iterations for our implementation, *vs.* 260k iterations in Chen et al. (2018a)) in Table. 7.1. During training, we resize the images to 512×512 pixels. This choice of resolution

Method	$\mathcal{G}$	res	GrabCut	Berkeley	ECSSD	MSRA-10K
TNS	✓	256	72.3 / 79.0	55.7 / 67.0	70.3 / 76.0	81.1 / 85.0
vgg-baseline	✗	256	73.5 / 77.4	58.2 / 63.2	71.2 / 72.3	83.4 / 86.2
vgg-early	✓	256	76.2 / 80.1	62.8 / 65.3	74.8 / 76.5	87.1 / 87.5
resnet-baseline	✗	256	81.6 / 83.0	68.5 / 68.2	80.2 / 82.0	86.4 / 86.9
resnet-early	✓	256	83.3 / 84.3	75.0 / 75.3	82.0 / 83.6	88.6 / 89.6
resnet-multi	✓	256	84.1 / 85.7	75.1 / 78.4	81.9 / 85.2	91.5 / 92.1
resnet-baseline	✗	512	76.1 / 79.0	65.5 / 68.3	79.9 / 82.6	87.0 / 87.9
resnet-early	✓	512	82.9 / 84.5	76.2 / 78.1	85.6 / 85.7	91.5 / 91.4
resnet-multi	✓	512	83.1 / 86.2	80.1 / 81.3	86.8 / 87.1	92.5 / 93.1

Table 7.1: Ablation Study: Tap-and-Shoot Segmentation (Chen et al., 2018a). 'res' refers to the image resolution used during training. We report average mIoU for the segmentation results after training for 16k iterations and after training convergence. The *-baseline* models receive a 3-channel RGB image as input without the guidance map $\mathcal{G}$. For comparison, we report the performance of (Chen et al., 2018a) in the first row (abbreviated as TNS).

is driven primarily by matching the resolution to that of the training images for the ResNet-101 backbone (He et al., 2016).

The *-baseline* models are trained using only the 3-channel RGB image and the instance ground truth mask without any user click transformations. The *-early* models use Gaussian guidance maps (Mahadevan et al., 2018); the network input is 3-channel RGB image and Gaussian encoding of the user's tap on the object of interest (Fig. 7.2(a)). The *-multi* models refer to the multi-stage fusion models with Gaussian encoding of user clicks. Note that we do not train a late-fusion model; standalone late-fusion models show inferior performance compared to their early-fusion counterparts (Rakelly et al., 2018).

From Table. 7.1, we observe that our trained network converges mostly within 16k iterations. For simplistic datasets such as MSRA10K (Cheng et al., 2014) and ECSSD (Shi et al., 2015), the *vgg-baseline* without user click transformation compares favourably with the approach of Chen et al. (2018a) at the same training resolution of 256×256. *resnet-baseline* models trained with 512 × 512 images significantly outperform (Chen et al., 2018a) reporting absolute mIoU gains of till 7% across the datasets. Based on this result alone, we conclude that one-click (and standard) interactive segmentation approaches should be benchmarked on more challenging datasets such as Pascal VOC 2012 (Everingham et al., 2010) and MS COCO (Lin et al., 2014), which feature cluttered scenes, multiple objects, occlusions and challenging lighting conditions (see Table. 7.3).

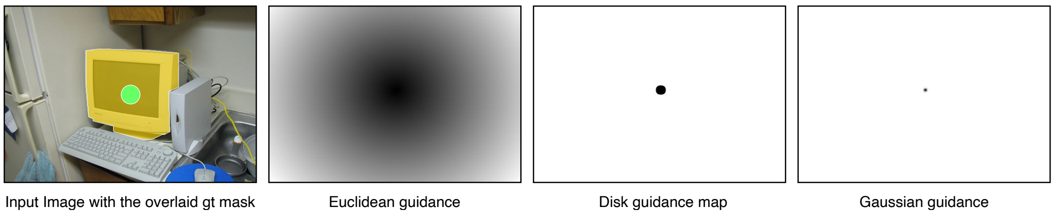


Figure 7.4: Examples of guidance maps. Given a click (shown as green circle) on the object of interest (highlighted in yellow), existing approaches transform it into guidance maps and uses it as an additional input channel. For ease of visualization, inverted values for the disk guidance map and the Gaussian guidance map are shown in the image.

Furthermore, with only the Gaussian transformation and ResNet-101 backbone trained on 512×512, we are able to achieve mIoU improvements in the range of 5-11% across datasets at convergence *w.r.t* Chen et al. (2018a). Having the multi-stage fusion offers us absolute mIoU gains of 1-4% w.r.t the early fusion variant (*resnet-early vs. resnet-multi* when trained with 512×512 images). Additionally, our *resnet* models require significantly less memory; 195.8 MB (stored as 32-bit/4-byte floating point numbers) instead of the 652.45 MB required for the segmentation network of Chen et al. (2018a).

7.4.3 Interactive image segmentation

Approaches in the literature (Xu et al., 2016; Mahadevan et al., 2018; Majumder and Yao, 2019a; Hu et al., 2019) are typically evaluated by (1) the average number of clicks needed to reach the desired level of segmentation (85% mIoU for Pascal VOC 2012, MS COCO, 90% mIoU for the less challenging Grabcut and Berkeley) and (2) the average mIoU *vs* the number of clicks.

The first criterion is primarily geared towards annotation tasks (Xu et al., 2016; Liew et al., 2017; Mahadevan et al., 2018; Jang and Kim, 2019; Sofiiuk et al., 2020) where high-quality segments are desired for each instance in the scene desirably with a low number of clicks. In this work, we are concerned primarily with achieving high-quality segments for the object of interest given only a single click. Accordingly, given a single user click, we report the average mIoU across all instances for the GrabCut, Berkeley and the Pascal VOC 2012 *val* dataset. For MS COCO object instances, following (Xu et al., 2016), we split the dataset into the 20 Pascal VOC 2012 categories and the 60 additional categories, and randomly sample 10 images per category for evaluation. We also report the average mIoU across these 800 sampled MS COCO instances for a fair comparison with Hu et al. (2019).

$\mathcal{G}$	GrabCut	Berkeley	VOC-2012	COCO-20	COCO-60
Euclidean	82.6	82.7	75.1	63.2	46.8
Disk	84.5	81.3	74.5	65.3	51.5
Gaussian	84.0	82.9	78.1	64.2	49.8
Gaussian-*multi*	**86.2**(2.2 ↑)	**84.0**(1.1 ↑)	**80.8**(2.7 ↑)	64.5(0.3 ↑)	**52.3**(2.5 ↑)

Table 7.2: We train separate networks to study the impact of different forms of user-click transformation - Euclidean (Xu et al., 2016), Gaussian (Benard and Gygli, 2017), and Disk (Benenson et al., 2019). The best results are indicated in **bold**. COCO-20 and COCO-60 refers to the instances from 20 overlapping categories and 60 non-overlapping categories of Pascal VOC 2012 respectively. The mIoU improvement (in %) when using the multi-stage framework is indicated using ↑.

For training (Xu et al., 2016; Majumder and Yao, 2019a; Hu et al., 2019), we use the ground truth masks of object instances from Pascal VOC 2012 (Everingham et al., 2010) *train* set with additional masks from Semantic Boundaries Dataset (SBD) (Hariharan et al., 2011) resulting in 10582 images. Note that unlike (Mahadevan et al., 2018), we do not use the training instances from MS COCO.

Ablation Study. We perform extensive ablation studies to thoroughly analyze the effectiveness of the individual components of our one-click segmentation framework. First, to validate our choice of guidance maps, we consider the user click transformations commonly used in existing interactive segmentation algorithms - Euclidean distance maps (Xu et al., 2016; Hu et al., 2019), Gaussian distance maps (Mahadevan et al., 2018) and disk (Benenson et al., 2019). Fig. 7.4 shows examples of such guidance maps. For each kind of guidance map, we train separate networks to understand the impact of different user click transformations. For evaluation, we report the average mIoU over all instances in the dataset, given a single click (see Table. 7.2). Next, we study the impact of our proposed late-fusion module (denoted by *-multi* in Table. 7.2); we observe an average mIoU improvement of around 1.8% across different datasets.

One-click segmentation. We next the segmentation performance of our method with existing interactive instance segmentation approaches (see Table. 7.3). The approaches are grouped separately into 3 different categories - pre-deep learning approaches, deep learning-based interactive instance segmentation approaches and tap-and-shoot segmentation approaches. From Table. 7.3, we observe that our approach outperforms the classical interactive segmentation works by a significant margin reporting 40% absolute improvement in average mIoU. We also outperform existing state-of-the-art interactive instance segmentation approaches by a considerable

Method	Network	GrabCut	Berkeley	VOC-2012	COCO-20	COCO-60
GC	-	41.7	33.8	27.7	-	8.9
GM	-	23.7	24.5	23.8	-	22.1
GD	-	48.8	36.1	31.0	-	25.2
iFCN	FCN-8s	62.9	61.3	53.6		**42.9**
ITIS	Deeplabv3+	82.1	-	71.0	-	-
CAG	FCN-8s	**83.2**	-	**74.0**	-	-
TS	FCN-8s	77.7	**74.5**	62.3	**42.5**	42.5
TNS	FCN-8s	79.0	67.0	-	-	-
Ours-best	Deeplabv2	**86.2**(3.0 ↑)	**84.0**(9.5 ↑)	**80.8**(6.8 ↑)	**64.5**(22.0 ↑)	**52.3**(9.6 ↑)

Table 7.3: Average mIoU given a single click. The approaches are grouped separately into 3 different categories - pre-deep learning approaches, deep learning-based interactive instance segmentation approaches and tap-and-shoot segmentation approaches respectively. For GC(Boykov and Jolly, 2001), GM(Bai and Sapiro, 2009), GD(Gulshan et al., 2010), and iFCN(Xu et al., 2016) we make use of the values provided by the authors of iFCN(Xu et al., 2016). We additionally use the mIoU values reported in ITIS (Mahadevan et al., 2018), CAG (Majumder and Yao, 2019a), the two-stream framework of Hu et al. (2019), and TNS (Chen et al., 2018a). The mIoU improvement (in %) over existing state-of-the-art approaches is indicated using ↑.

margin ($> 3\%$). Additionally, we report an absolute mIoU improvement of 7.2% and 17% on Grabcut and Berkeley over the tap-and-shoot segmentation framework of Chen et al. (2018a). We show qualitative results to demonstrate the effectiveness of our proposed algorithm (see Fig. 7.5). The resulting segmentations demonstrate that our approach is highly effective for the one-click segmentation paradigm.

7.5 User Study

Across existing state-of-the-art interactive frameworks (Xu et al., 2016; Mahadevan et al., 2018; Sofiiuk et al., 2020), user clicks are simulated following the protocols established in (Xu et al., 2016; Mahadevan et al., 2018). For our user study, we consult 5 participants uninitiated to the task of interactive segmentation. We prepare a toy dataset with 50 object instances from the MSRA10K (Cheng et al., 2014) dataset. We presented the image with the segmentation mask for the target object overlaid on the image and asked the users to provide their click.

During training, we applied random perturbations of $\mathcal{U}(-50, 50)$ pixels to the center of mass of the object instance to obtain the final user click. Our user study found that participants placed clicks at a mean distance of 24 pixels from the center of the mask with a standard deviation of 27 pixels. This result validates our assumption

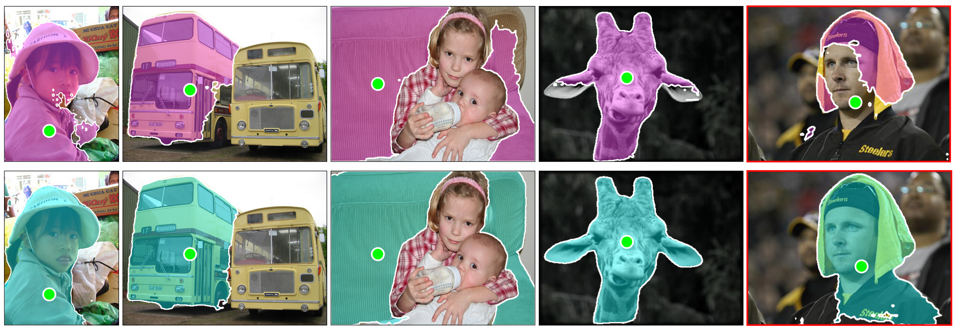


Figure 7.5: Qualitative Results. Incorporating the user clicks at different stages of the network leads to an improvement in the quality of masks generated (second row) *w.r.t* the early-fusion variants (first row). Click locations are shown in green circles. The extreme right column shows a scenario where both the networks failed to generate a satisfactory mask.

that users are more likely to click in the vicinity of the object's center-of-mass. It also supports our click sampling scheme for generating training instances when training the object selection stage. On average, we observed that users took 2.3 seconds with a standard deviation of 0.8 seconds to position their click.

7.6 Conclusion

In this work, we propose a one-click segmentation framework that produces high-quality segmentation masks. We validated our design choices through detailed ablation studies; we observed that having a multi-stage module improves the segmentation framework and gives the network an edge over its early-fusion variants. Via experiments, we observed that for the single click scenario, our proposed approach significantly outperforms existing state-of-the-art approaches - including the more complicated interactive instance segmentation models using state-of-the-art segmentation models (Chen et al., 2018c).

However, we observe existing tap-and-shoot segmentation frameworks (Chen et al., 2018a), including our proposed framework, are limited by their inability to learn from negative clicks (Xu et al., 2016; Mahadevan et al., 2018). One major drawback of such a training scenario is that the network does not have a notion of corrective clicking; if the generated segmentation mask extends beyond the object boundaries, it cannot rectify this mistake. Clicking on locations outside the object can mitigate this effect, though this then deviates from tap-and-shoot interaction.

Chapter 8

Conclusion

8.1 Overview

In the future, as deep learning models increase in size, the need for large-scale, high-quality annotations will only rise. To ensure the steady progress of deep learning segmentation models, we require interactive frameworks to scale up with this increasing need. In this dissertation, we propose different interactive frameworks with the goal of efficiently leveraging user-provided clicks as we strive towards the shared goal of one-click segmentation. In this final chapter, we summarize our contributions to interactive segmentation research. Finally, we conclude the chapter and the dissertation discussing the challenges often encountered when designing interactive frameworks and what holds in the future for interactive image segmentation.

8.2 Contributions

Through this dissertation, we highlighted and addressed different facets of interactive instance segmentation. We summarize our contribution as follows:

User click representation. Across several interactive frameworks (Xu et al., 2016; Liew et al., 2017; Li et al., 2018; Jang and Kim, 2019; Sofiiuk et al., 2020), users interact with the segmentation models via pixel clicks. These clicks are far and few, e.g., at the onset of interaction, users mark only a single pixel on the target object. Information encoded in such clicks is sparse. Accordingly, existing approaches apply distance-based transformations to de-sparsify the information such that segmentation backbones can meaningfully leverage them. These distance-based transformations include Euclidean (Xu et al., 2016; Li et al., 2018) and Gaussian (Mahadevan et al., 2018; Jang and Kim, 2019; Sofiiuk et al., 2020) distance-based transformation. As research in the field progressed, the role of user-click transformations in the performance of interactive frameworks went un-noticed. Following works focused primarily on adapting contemporary state-of-the-art FCN architectures (Mahadevan et al., 2018) and iterative training procedures (Liew et al., 2017; Mahadevan et al., 2018).

We observed that existing guidance maps disregard basic consistencies in the scene, such as color, textures (Achanta et al., 2012), and object proposals (Pont-Tuset et al., 2017) that can be obtained in an unsupervised manner. By meaningfully leveraging such low-level cues (Yao et al., 2015; Achanta et al., 2012) and high-level cues (Pont-Tuset et al., 2017; Maninis et al., 2016), we proposed an effective user-click transformation framework that significantly impacts the performance of interactive frameworks. Our work was the first in deep interactive segmentation literature to investigate and highlight the impact of guidance maps in the performance of interactive frameworks. Using an outdated segmentation backbone FCN-8s (Long et al., 2015), we were able to comprehensively outperform contemporary interactive frameworks using state-of-the-art segmentation backbones such as Deeplabv3+ (Chen et al., 2018c) across all public benchmarks.

Localized interactions. Intuitively, a user interested in segmenting an object instance from the scene would primarily focus in the vicinity of object. User clicks, thereby, should either be on the object or close to object. However, across existing frameworks (Liew et al., 2017; Li et al., 2018; Sofiiuk et al., 2020) including our work (Majumder and Yao, 2019a), user-interactions are generated synthetically following heuristics outlined in (Xu et al., 2016). While positive clicks are still restricted, by their definition, to be within the object confinements, negative clicks are allowed anywhere on the background pixels even for pixels far apart from the object. Other frameworks which restrict the placement of user clicks do so via enforcing hard constraints (Xu et al., 2017b; Maninis et al., 2018); such frameworks are typically non-iterative and users cannot refine the errors segmentation masks (Xu et al., 2017b).

Our motivation was to direct the user clicks to the vicinity of the object without using hard constraints, e.g., extreme clicks (Maninis et al., 2018; Agustsson et al., 2019). Spatially limiting user clicks gives us the opportunity to enforce a *weak* constraint on the location and scale of the object which in turn facilitates the network training under less ambiguous circumstances. In our work, we proposed a new transformation scheme for the user-provided clicks which provides a weak localization prior on the object of interest and while being consistent with scene-level consistencies such as edges, textures, etc. in the scene.

Two-in-one refinement. Instead of conventional positive and negative clicks to fix false negatives and false positives (Xu et al., 2016; Liew et al., 2017; Mahadevan et al., 2018; Majumder and Yao, 2019a; Sofiiuk et al., 2020), we ask users to click on object boundaries in the vicinity of largest segmentation errors. This lessens the effort on the user's part. Previously, the user had to identify and then click centrally in the largest erroneous regions. Instead, our proposed boundary refinement clicks is intuitive for the user. Additionally, it provides a stronger cue for the object location. Unlike our previous works (Majumder and Yao, 2019a,b), we use Gaussians for user-click transformation. Content-aware user-click encodings do offer significant benefit over Gaussians and Euclidean transforms, as we demonstrated in previous chapters. However object proposals (Pont-Tuset et al., 2017) necessary for such encodings are expensive to obtain; processing a single image can take tens of seconds. Secondly, from an implementation stand-point, Gaussian encodings are easier to use.

Also, current frameworks treat all positive clicks indiscriminately (Xu et al., 2016; Liew et al., 2017; Mahadevan et al., 2018; Majumder and Yao, 2019a; Sofiiuk et al., 2020). However, in practice, the network leverages the first positive click to select the object. Following positive clicks are used to mark false negative errors. Our work was the first to frame interactive segmentation as a two stage task - select and refine. Through experiments, we demonstrate the benefits of a two-stage approach and we believe that disentangling the tasks will enable the research community to develop streamlined networks for the selection and refinement task.

Multi-stage fusion of user interaction. In this work, we aim for maximizing the performance achievable with a single user click. The end goal is to get closer to one-click segmentation - the endgame for interactive frameworks. Having a good initial segmentation mask is crucial for several commercial downstream approaches such as Photoshop and mobile photography (Chen et al., 2018a). Despite significant progress over the years, advanced deep learning-based frameworks tend to under-utilize and particularly fare poorly with only one or two clicks.

To make the most of first (few) click(s), we propose the fusion of user guidance at multiple stages of the network rather than via early fusion. A late fusion allows the user interaction to have a direct and more pronounced effect on the final segmentation mask. While previous approaches attempted late fusion of user interaction (Hu et al., 2019; Rakelly et al., 2018), it came with a heavy cost with newer networks requiring additional parameters in the order of 100 million (Rakelly et al., 2018). Instead, we propose an light-weight and easier-to-train alternative to perform fusion of user interactions at multiple stages of the network. For a single click, our design outperforms all competing approaches with minimal computational overhead.

Summary. User clicks drive interactive segmentation; they provide cues on the object location and the segmentation error. However, clicks are expensive and difficult to obtain. Our goal is thus to make the most out of every user-provided click and build click-efficient interactive frameworks. We explore the design of such efficient frameworks by extending the capabilities of different components in the interactive pipeline - user click representation, click refinement, and network design. First, to make clicks more informative, we leverage image features and click locations to provide additional image content awareness and spatial context to user clicks. Next, we bring more utility to the user-provided clicks by proposing boundary clicks for refinement. Finally, we propose a network design that guarantees user clicks have more impact on the segmentation output.

8.3 Discussions and Outlook

User study. Existing interactive frameworks are designed on the premise of integrating and collaborating with actual user inputs to speed up dense (e.g., pixel-level) annotation of images. Ideally, for training such frameworks, real user input is desired. However, such annotations are expensive to acquire (Kass et al., 1988; Xu et al., 2016; Khoreva et al., 2017). Instead, training and test clicks are synthesized using instance-level masks present in the dataset. Typically, most approaches use the click sampling strategies established in (Xu et al., 2016) and (Mahadevan et al., 2018); other methods synthesize clicks per their algorithm (Liew et al., 2019; Sofiiuk et al., 2020). Assumptions made in such cases might not hold, limiting the methods' performance when interacting with an actual user.

Furthermore, click strategies might vary in difficulty to the user even when they serve the same functionality. For example, tight bounding boxes and extreme clicks both enclose an object of interest with tight boundaries. However, users spend an average of 7 seconds (Papadopoulos et al., 2017; Maninis et al., 2018) to mark the four extremities of an object; for bounding boxes it takes an user an average of 35 seconds (Su et al., 2012; Papadopoulos et al., 2017).

Currently, only a handful of click-based interactive methods (Liew et al., 2019; Majumder et al., 2020b,a) take user-clicking behavior into accounts. To ensure that our frameworks (Majumder et al., 2020b,a) are consistent with the users, we performed user studies to validate our click sampling strategies. In the future, newer interactive frameworks should validate their click sampling strategies using actual users and also compare the merits and demerits of a newly proposed click sampling strategy for the user.

Datasets and Evaluation. Deep learning-based interactive frameworks rely on dense pixel-level annotation for object instances during training. The commonly used datasets for training are Pascal VOC 2012 (Everingham et al., 2010) and the MS COCO (Lin et al., 2014) *train* set. These datasets capture the diversity of unconstrained imagery and have been pivotal to the success of current state-of-the-art methods in semantic (Long et al., 2015; Chen et al., 2018c), instance (He et al., 2017), and interactive segmentation algorithms (Xu et al., 2016).

Current state-of-the-art interactive methods achieve overall good segmentation performance; however, due to the coarse nature of the annotation, and inconsisten-

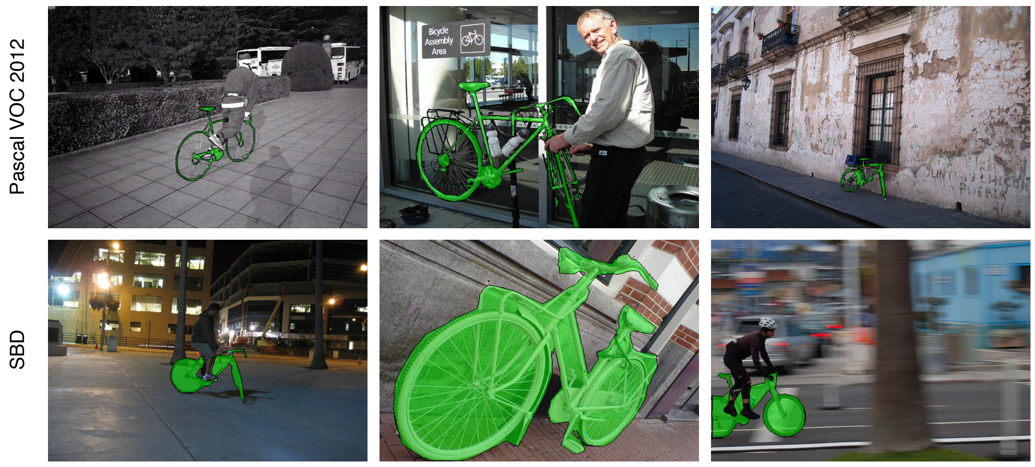


Figure 8.1: We show some examples demonstrating the lack of consistency in the per-pixel annotation of the instances from the 'bicycle' class in the Pascal VOC 2012 (Everingham et al., 2010) and SBD (Hariharan et al., 2011). However, despite this inconsistency, several interactive frameworks rely on the annotations from both these datasets jointly during training. The ground truth masks are overlaid in green.

cies across datasets, instances from certain object classes incur heavy penalties in the form of additional clicks. Thin object segmentation, i.e. segmentation of object instances with elongated thin structures, is one challenging aspect where current methods falter. To circumvent this, several approaches (Li et al., 2018; Jang and Kim, 2019; Sofiiuk et al., 2020) opt for consistency in annotations. Instead of training on Pascal VOC 2012 *train* set with additional masks from SBD (Hariharan et al., 2011) which results in conflicting annotations (Fig. 8.1), these methods opt to train and validate only on the SBD *train* and *val* set. Recently, the work of (Liew et al., 2021) proposed the ThinObject-5K dataset with high quality annotations for segmenting objects with *thin* regions and proposed a framework targeting the segmentation of objects of such nature.

Another challenging aspect is small objects, i.e. objects occupying less than 32×32 pixels (Noh et al., 2015). Segmentation of small objects is challenging due to the inherent limitations of CNNs. This is further exacerbated by the imbalance in the datasets when it comes to small *vs* large objects (Majumder and Yao, 2019a). To address the dataset imbalance, several current methods rely on class-balanced cross entropy loss functions (Maninis et al., 2018; Majumder and Yao, 2019a). Additionally, certain approaches (Li et al., 2018; Liew et al., 2019) including ours (Majumder and Yao, 2019a) adopt scale-aware measures to ensure the quality of masks generated for small objects.

Ideally, developed interactive frameworks should be robust against variations in object sizes and appearance. However, current benchmarks do not distinguish between the object diversity. In such cases, the gains in segmenting larger objects are offset by the heavy click penalties incurred by small objects and objects with thin structures. Going ahead, interactive frameworks should report the performance on small, large and thin objects separately to underline the versatility of their proposed approach.

Interaction modality. Throughout the dissertation, most interactive frameworks in discussion made use of user input in the form of spatial interactions, e.g. bounding boxes (Xu et al., 2017b; Agustsson et al., 2019), polygon vertices (Castrejon et al., 2017; Acuna et al., 2018), and pixel clicks (Xu et al., 2016; Liew et al., 2017; Jang and Kim, 2019; Sofiiuk et al., 2020). Spatial interactions are easily obtainable and provide the segmentation backbone *with* the knowledge of *where* the object is. However, these spatial interactions can fall short in instances where there are diverse scales of object candidates at the clicked user location (Li et al., 2018; Lombaert et al., 2005). Natural language phrases, in such cases, can guide the segmentation network by explicitly specifying *what* the target of interest is (Fig. 8.2).

Although language-based inputs have been previously considered in literature for the task of automated image segmentation (Hu et al., 2016), the use of such inputs are limited in interactive segmentation literature. To date, only the work of Ding et al. (2020) considers the use of language-based inputs for the task of interactive segmentation. In future, language-based inputs could be used for better representation of user intention when segmenting the target object.

Future direction. Interactive frameworks have improved leaps and bounds over the last few years. Even for challenging datasets such as Pascal VOC 2012 (Everingham et al., 2010), the number of clicks required to reach 85% mIoU has reduced drastically from 15.06 (Rother et al., 2004) to 2.88 (Majumder et al., 2020b). For simpler datasets such as the GrabCut dataset (Rother et al., 2004), it is now possible to obtain segmentation masks with 90% *w.r.t* ground truth masks with around 2 clicks. The goal of one-click segmentation certainly seems within reach.

Human-machine collaborative annotation and commercial applications are typically mentioned as downstream applications across most interactive frameworks (Mahadevan et al., 2018; Andriluka et al., 2018; Li et al., 2018). However, the difference in requirements of these downstream applications are often overlooked. Instead, most interactive frameworks, typically target the 85%-90% mIoU *w.r.t* ground truth masks of the instances (Xu et al., 2016). While a mIoU in the range

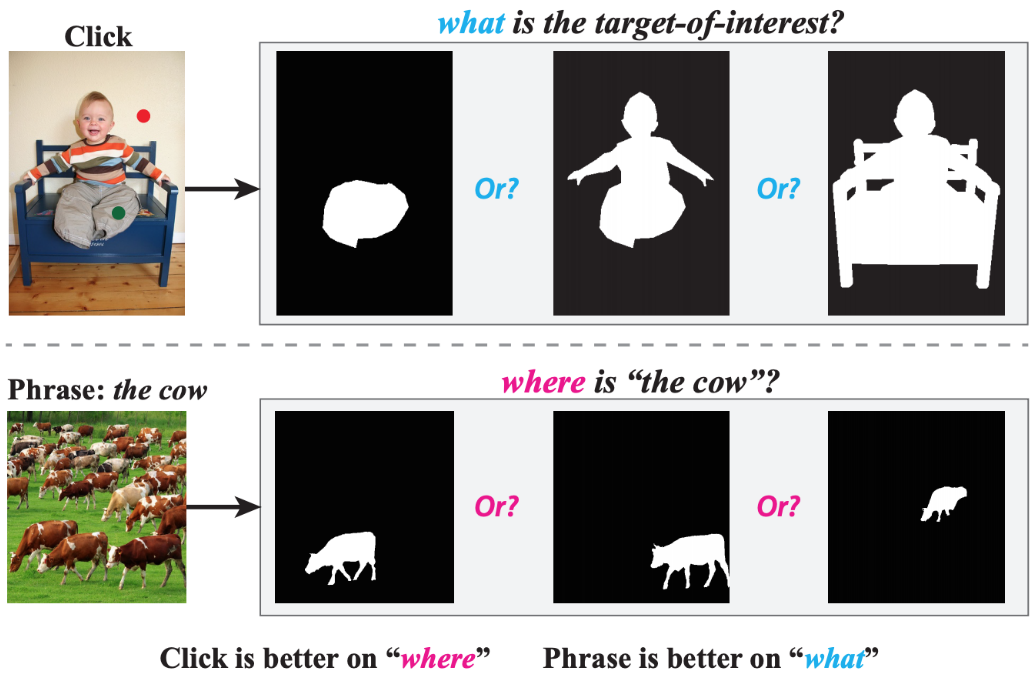

Click
what is the target-of-interest?
Or?
Or?
Phrase: the cow
where is "the cow"?
Or?
Or?
Click is better on "where"
Phrase is better on "what"

Figure 8.2: Clicks and phrases have their own advantages and dis-advantages. For interactive frameworks, spatial interactions, e.g. clicks, are more suited for providing cues on where the object is. Phrases, on the other hand, can help interactive frameworks by disambiguating among competing instances at the user-provided clicked location. Image from Ding et al. (2020).

of 85%-90% is satisfactory for annotation purposes, for downstream applications such as Photoshop, a mIoU of 99% would certainly be more desirable than a 90% mIoU (Forte et al., 2020). Besides, outputs with high mIoU *w.r.t* ground truth masks, e.g. 95% on the SBD dataset (Hariharan et al., 2011), does not guarantee high quality masks as ground truth masks themselves might not follow the object boundaries (Forte et al., 2020). Furthermore, for commercial software and applications such as image matting (Xu et al., 2017a), the low quality of the images in the current datasets are prohibitive. A *one-size-fits-all* approach to interactive framework design can thus be counter-productive. Instead, in future, design of interactive frameworks need to be application driven.

In recent years, interactive image frameworks have been integral to the progress in deep learning-based interactive video object segmentation approaches. Interestingly, in such interactive video object segmentation (VOS) frameworks (Cheng et al., 2021), the first step is similar to interactive image segmentation; the user annotates one of the frames with clicks (Cheng et al., 2021) or scribbles (Oh et al.,

2019) which is used to generate the object mask. This generated mask is then propagated across the entire video (Cheng et al., 2021; Oh et al., 2019). With videos becoming ubiquitous, the demand for *stronger* video understanding models will drive the need for densely annotated video datasets. Consequently, over the course of next few years, research in interactive frameworks for images and videos will receive significant attention. Interactive image segmentation has been an integral part of computer vision research for decades (Kass et al., 1988; Rother et al., 2004; Lin et al., 2020) and will continue to do so in future (Andriluka et al., 2020).

Bibliography

Achanta, R., Shaji, A., Smith, K., Lucchi, A., Fua, P., Süsstrunk, S., et al. (2012). SLIC superpixels compared to state-of-the-art superpixel methods. *TPAMI*, 34(11):2274–2282. xi, 9, 39, 40, 46, 47, 50, 55, 58, 75, 107, 108

Acuna, D., Ling, H., Kar, A., and Fidler, S. (2018). Efficient interactive annotation of segmentation datasets with polygon-rnn++. In *CVPR*, pages 859–868. x, 3, 4, 5, 29, 112

Agustsson, E., Uijlings, J. R., and Ferrari, V. (2019). Interactive full image segmentation by considering all regions jointly. In *CVPR*, pages 11622–11631. viii, x, 7, 15, 27, 28, 81, 108, 112

Andriluka, M., Pellegrini, S., Popov, S., and Ferrari, V. (2020). Efficient full image interactive segmentation by leveraging within-image appearance similarity. *arXiv preprint arXiv:2007.08173*. 27, 28, 114

Andriluka, M., Uijlings, J. R., and Ferrari, V. (2018). Fluid annotation: A human-machine collaboration interface for full image annotation. In *MM*. 4, 5, 27, 28, 46, 65, 79, 112

Arbelaez, P., Maire, M., Fowlkes, C., and Malik, J. (2011). Contour detection and hierarchical image segmentation. *TPAMI*, 33(5):898–916. 52

Audebert, N., Boulch, A., Le Saux, B., and Lefèvre, S. (2019). Distance transform regression for spatially-aware deep semantic segmentation. *CVIU*, 189(102809). 84

Bai, J. and Wu, X. (2014). Error-tolerant scribbles based interactive image segmentation. In *CVPR*, pages 392–399. 1

Bai, X. and Sapiro, G. (2009). Geodesic matting: A framework for fast interactive image and video segmentation and matting. *IJCV*, 82(2):113–132. xiv, xvii, xix, 13, 48, 59, 67, 90, 91, 105

Ballard, D. H. and Brown, C. M. (1982). Computer vision. englewood cliffs. *J: Prentice Hall*. 1

Barrett, W. A. and Mortensen, E. N. (1996). Fast, accurate, and reproducible live-wire boundary extraction. In *International Conference on Visualization in Biomedical Computing*, pages 183–192. Springer. 16

Barrett, W. A. and Mortensen, E. N. (1997). Interactive live-wire boundary extraction. *Medical image analysis*, 1(4):331–341. 13, 16

Bearman, A., Russakovsky, O., Ferrari, V., and Fei-Fei, L. (2016). What's the point: Semantic segmentation with point supervision. In *ECCV*, pages 549–565. Springer. 3

Benard, A. and Gygli, M. (2017). Interactive video object segmentation in the wild. *arXiv preprint:1801.00269*. xvii, xviii, 5, 6, 7, 8, 9, 13, 14, 24, 46, 47, 48, 49, 54, 59, 65, 66, 68, 69, 70, 74, 77, 78, 79, 81, 83, 84, 85, 87, 88, 104

Benenson, R., Popov, S., and Ferrari, V. (2019). Large-scale interactive object segmentation with human annotators. In *CVPR*, pages 11700–11709. xviii, 3, 4, 5, 7, 8, 24, 65, 68, 71, 79, 81, 82, 100, 104

Blake, A. and Isard, M. (1998). *Active shape models.* Springer London. 15

Boykov, Y. and Kolmogorov, V. (2004). An experimental comparison of min-cut/max-flow algorithms for energy minimization in vision. *IEEE transactions on pattern analysis and machine intelligence*, 26(9):1124–1137. 17, 18

Boykov, Y. Y. and Jolly, M.-P. (2001). Interactive graph cuts for optimal boundary & region segmentation of objects in nd images. In *ICCV*. xiv, xvii, xix, 1, 2, 13, 17, 18, 20, 46, 48, 59, 67, 90, 91, 105

Breiman, L. (2001). Random forests. *Machine learning*, 45(1):5–32. 14

Cadena, C. and Košecká, J. (2014). Semantic segmentation with heterogeneous sensor coverages. In *ICRA*, pages 2639–2645. IEEE. 39

Caelles, S., Maninis, K.-K., Pont-Tuset, J., Leal-Taixé, L., Cremers, D., and Van Gool, L. (2017). One-shot video object segmentation. In *CVPR*, pages 221–230. 78

Calivá, F., Iriondo, C., Martinez, A., Majumdar, S., and Pedoia, V. (2019). Distance map loss penalty term for semantic segmentation. *arXiv preprint arXiv:1908.03679*. 84

Carreira, J. and Sminchisescu, C. (2011). CPMC: Automatic object segmentation using constrained parametric min-cuts. *TPAMI*, (7):1312–1328. 40, 55

Castrejon, L., Kundu, K., Urtasun, R., and Fidler, S. (2017). Annotating object instances with a polygon-rnn. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5230–5238. 3, 4, 5, 29, 82, 112

Chen, D.-J., Chien, J.-T., Chen, H.-T., and Chang, L.-W. (2018a). Tap and shoot segmentation. In *AAAI*. xviii, xix, 1, 11, 48, 60, 69, 71, 74, 82, 85, 93, 96, 100, 101, 102, 103, 105, 106, 109

Chen, L.-C., Papandreou, G., Kokkinos, I., Murphy, K., and Yuille, A. L. (2018b). Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. *TPAMI*, 40(4):834–848. xi, xvii, 2, 5, 6, 31, 36, 38, 42, 59, 74, 81, 83, 84, 87, 95, 96, 98

Chen, L.-C., Zhu, Y., Papandreou, G., Schroff, F., and Adam, H. (2018c). Encoder-decoder with atrous separable convolution for semantic image segmentation. In *ECCV*. xvii, 2, 3, 5, 6, 42, 43, 59, 67, 81, 84, 87, 95, 96, 106, 108, 110

Cheng, H. K., Tai, Y.-W., and Tang, C.-K. (2021). Modular interactive video object segmentation: Interaction-to-mask, propagation and difference-aware fusion. In *CVPR*, pages 5559–5568. 113, 114

Cheng, M.-M., Mitra, N. J., Huang, X., Torr, P. H., and Hu, S.-M. (2014). Global contrast based salient region detection. *IEEE TPAMI*, 37(3):569–582. 96, 101, 102, 105

Cohen, L. D. and Cohen, I. (1993). Finite-element methods for active contour models and balloons for 2-d and 3-d images. *PAMI*, 15(11):1131–1147. 15

Cordts, M., Omran, M., Ramos, S., Rehfeld, T., Enzweiler, M., Benenson, R., Franke, U., Roth, S., and Schiele, B. (2016). The cityscapes dataset for semantic urban scene understanding. In *CVPR*. vii, 1, 2, 3, 5, 79

Cremers, D., Tischhäuser, F., Weickert, J., and Schnörr, C. (2002). Diffusion snakes: Introducing statistical shape knowledge into the mumford-shah functional. *IJCV*, 50(3):295–313. 13, 15

Criminisi, A., Sharp, T., and Blake, A. (2008). Geos: Geodesic image segmentation. In *ECCV*. 13, 46, 48, 68

Dai, J., He, K., and Sun, J. (2015). Boxsup: Exploiting bounding boxes to supervise convolutional networks for semantic segmentation. In *ICCV*. 3, 52

Dai, J., He, K., and Sun, J. (2016). Instance-aware semantic segmentation via multi-task network cascades. In *CVPR*. 4, 46

Das, P., Veksler, O., Zavadsky, V., and Boykov, Y. (2006). Semiautomatic segmentation with compact shape prior. *Canadian Conference on Computer and Robot Vision*, 27(1-2):28–36. 19

Davis, L. S. (1975). A survey of edge detection techniques. *Computer graphics and image processing*, 4(3):248–270. 1

Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., and Fei-Fei, L. (2009). Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee. 14

Ding, H., Cohen, S., Price, B., and Jiang, X. (2020). Phraseclick: toward achieving flexible interactive segmentation by phrase and click. In *ECCV*, pages 417–435. Springer. xvi, 112, 113

Dollár, P. and Zitnick, C. L. (2013). Structured forests for fast edge detection. In *ICCV*, pages 1841–1848. 41

Doyle, W. (1962). Operations useful for similarity-invariant pattern recognition. *Journal of the ACM (JACM)*, 9(2):259–267. 1

Dutt Jain, S. and Grauman, K. (2013). Predicting sufficient annotation strength for interactive foreground segmentation. In *ICCV*, pages 1313–1320. 71

Endres, I. and Hoiem, D. (2013). Category-independent object proposals with diverse ranking. *IEEE PAMI*, 36(2):222–234. 40

Everingham, M., Van Gool, L., Williams, C. K., Winn, J., and Zisserman, A. (2010). The pascal visual object classes (voc) challenge. *IJCV*, 88(2):303–338. xv, 4, 5, 13, 14, 22, 41, 42, 43, 47, 54, 56, 72, 74, 81, 82, 83, 86, 87, 88, 89, 101, 102, 104, 110, 111, 112

Faktor, A. and Irani, M. (2014). Video segmentation by non-local consensus voting. In *BMVC*. 50, 70

Falcao, A. X., Udupa, J. K., Samarasekera, S., Sharma, S., Hirsch, B. E., and Lotufo, R. d. A. (1998). User-steered image segmentation paradigms: Live wire and live lane. *Graphical models and image processing*, 60(4):233–260. 13, 16, 82

Ford Jr, L. R. and Fulkerson, D. R. (1962). *Flows in networks*. Princeton university press. 17

Forte, M., Price, B., Cohen, S., Xu, N., and Pitié, F. (2020). Getting to 99% accuracy in interactive segmentation. *arXiv preprint arXiv:2003.07932*. 113

Freedman, D. and Zhang, T. (2005). Interactive graph cut based segmentation with shape priors. In *CVPR*, volume 1, pages 755–762. IEEE. 19

Fu, K.-S. and Mui, J. (1981). A survey on image segmentation. *Pattern recognition*, 13(1):3–16. 1

Fukushima, K. and Miyake, S. (1982). Neocognitron: A self-organizing neural network model for a mechanism of visual pattern recognition. In *Competition and cooperation in neural nets*, pages 267–285. Springer. 13

Gleicher, M. (1995). Image snapping. In *SIGGRAPH*. 82

Goldberg, A. V. and Tarjan, R. E. (1988). A new approach to the maximum-flow problem. 17

Goodfellow, I., Bengio, Y., and Courville, A. (2016). *Deep learning*. MIT press. 32, 34

Gould, S., Rodgers, J., Cohen, D., Elidan, G., and Koller, D. (2008). Multi-class segmentation with relative location prior. *IJCV*, 80(3):300–316. 39

Grady, L., Schiwietz, T., Aharon, S., and Westermann, R. (2005). Random walks for interactive organ segmentation in two and three dimensions: Implementation and validation. In *MICCAI*. xiv, xvii, 13, 18, 21, 46, 59, 65, 90, 91

Gulshan, V., Rother, C., Criminisi, A., Blake, A., and Zisserman, A. (2010). Geodesic star convexity for interactive image segmentation. In *CVPR*. xvii, xix, 13, 19, 48, 59, 68, 96, 105

Hafiz, A. M. and Bhat, G. M. (2020). A survey on instance segmentation: state of the art. *International Journal of Multimedia Information Retrieval*, pages 1–19. 1

Hao, S., Zhou, Y., and Guo, Y. (2020). A brief survey on semantic segmentation with deep learning. *Neurocomputing*, 406:302–321. 1

Hariharan, B., Arbelaez, P., Bourdev, L., Maji, S., and Malik, J. (2011). Semantic contours from inverse detectors. In *ICCV*. xv, 43, 54, 72, 74, 86, 104, 111, 113

Hariharan, B., Arbeláez, P., Girshick, R., and Malik, J. (2014). Simultaneous detection and segmentation. In *ECCV*. 46

Hariharan, B., Arbeláez, P., Girshick, R., and Malik, J. (2015). Hypercolumns for object segmentation and fine-grained localization. In *CVPR*. 46

He, K., Gkioxari, G., Dollár, P., and Girshick, R. (2017). Mask R-CNN. In *ICCV*. x, 2, 4, 21, 27, 28, 36, 46, 85, 110

He, K., Zhang, X., Ren, S., and Sun, J. (2016). Deep residual learning for image recognition. In *CVPR*. x, 2, 14, 31, 34, 35, 36, 66, 74, 76, 83, 98, 102

He, X., Zemel, R. S., and Ray, D. (2006). Learning and incorporating top-down cues in image segmentation. In *ECCV*. 50, 70

Hearst, M. A., Dumais, S. T., Osuna, E., Platt, J., and Scholkopf, B. (1998). Support vector machines. *IEEE Intelligent Systems and their applications*, 13(4):18–28. 14

Heo, Y., Jun Koh, Y., and Kim, C.-S. (2020). Interactive video object segmentation using global and local transfer modules. In *ECCV*, pages 297–313. Springer. 27

Heo, Y., Koh, Y. J., and Kim, C.-S. (2021). Guided interactive video object segmentation using reliability-based attention maps. In *CVPR*, pages 7322–7330. 27

Hinton, G. E., Srivastava, N., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. R. (2012). Improving neural networks by preventing co-adaptation of feature detectors. *arXiv preprint arXiv:1207.0580*. 14

Holschneider, M., Kronland-Martinet, R., Morlet, J., and Tchamitchian, P. (1990). A real-time algorithm for signal analysis with the help of the wavelet transform. In *Wavelets*, pages 286–297. Springer. xi, 38

Hu, J., Shen, L., and Sun, G. (2018). Squeeze-and-excitation networks. In *CVPR*, pages 7132–7141. xv, 98, 99

Hu, R., Rohrbach, M., and Darrell, T. (2016). Segmentation from natural language expressions. In *European Conference on Computer Vision*, pages 108–124. Springer. 112

Hu, Y., Soltoggio, A., Lock, R., and Carter, S. (2019). A fully convolutional two-stream fusion network for interactive image segmentation. *Neural Networks*, 109:31–42. ix, xii, xv, xvii, xviii, xix, 9, 10, 11, 25, 45, 46, 47, 59, 60, 65, 68, 69, 77, 80, 82, 85, 88, 95, 96, 97, 99, 103, 104, 105, 109

Ioffe, S. and Szegedy, C. (2015). Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *ICML*, pages 448–456. PMLR. 34

Isack, H., Gorelick, L., Ng, K., Veksler, O., and Boykov, Y. (2018). K-convexity shape priors for segmentation. In *ECCV*, pages 36–51. 19

Jain, A. K. and Dubes, R. C. (1988). *Algorithms for Clustering Data*. Prentice-Hall, Inc., USA. 1

Jain, S. D. and Grauman, K. (2016). Active image segmentation propagation. In *CVPR*, pages 2864–2873. 3

Jang, W.-D. and Kim, C.-S. (2019). Interactive image segmentation via backpropagating refinement scheme. In *CVPR*, pages 5297–5306. xviii, 4, 7, 10, 25, 26, 42, 77, 80, 81, 82, 83, 87, 89, 90, 100, 103, 107, 111, 112

Kass, M., Witkin, A., and Terzopoulos, D. (1988). Snakes: Active contour models. *IJCV*, 1(4):321–331. viii, 1, 2, 13, 14, 15, 16, 17, 18, 26, 48, 67, 82, 110, 114

Khoreva, A., Benenson, R., Hosang, J., Hein, M., and Schiele, B. (2017). Simple does it: Weakly supervised instance and semantic segmentation. In *CVPR*, pages 876–885. 3, 13, 52, 110

Kontogianni, T., Gygli, M., Uijlings, J., and Ferrari, V. (2020). Continuous adaptation for interactive object segmentation by learning from corrections. In *ECCV*, pages 579–596. Springer. xviii, 80, 81, 82, 86, 87, 90

Krähenbühl, P. and Koltun, V. (2014). Geodesic object proposals. In *ECCV*. 52

Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. *NIPS*, 25:1097–1105. 14, 34, 35

Le, H., Mai, L., Price, B., Cohen, S., Jin, H., and Liu, F. (2018). Interactive boundary prediction for object selection. In *Proceedings of the ECCV (ECCV)*, pages 18–33. viii, 15, 24, 82

LeCun, Y., Bottou, L., Bengio, Y., and Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324. 14

Lempitsky, V. S., Kohli, P., Rother, C., and Sharp, T. (2009). Image segmentation with a bounding box prior. In *ICCV*. Citeseer. 19

Li, Y., Sun, J., Tang, C.-K., and Shum, H.-Y. (2004). Lazy snapping. *ACM Transactions on Graphics (ToG)*, 23(3):303–308. viii, ix, 1, 5, 13, 15, 17, 18, 20, 26, 46, 48, 65, 67, 82

Li, Z., Chen, Q., and Koltun, V. (2018). Interactive image segmentation with latent diversity. In *CVPR*. xii, xvii, xviii, 4, 5, 9, 10, 25, 42, 45, 46, 48, 58, 59, 60, 68, 77, 80, 82, 87, 98, 107, 108, 111, 112

Liang-Chieh, C., Papandreou, G., Kokkinos, I., Murphy, K., and Yuille, A. (2015). Semantic image segmentation with deep convolutional nets and fully connected crfs. In *ICLR*. xi, 2, 6, 38, 42, 52, 59

Liew, J., Wei, Y., Xiong, W., Ong, S.-H., and Feng, J. (2017). Regional interactive image segmentation networks. In *ICCV*. viii, ix, xii, xiv, xvii, xviii, 4, 5, 7, 9, 10, 15, 23, 24, 25, 42, 45, 46, 47, 48, 49, 51, 54, 59, 65, 66, 68, 69, 72, 74, 77, 80, 81, 82, 83, 86, 87, 88, 89, 90, 91, 94, 98, 100, 103, 107, 108, 109, 112

Liew, J. H., Cohen, S., Price, B., Mai, L., and Feng, J. (2021). Deep interactive thin object selection. In *WACV*, pages 305–314. 111

Liew, J. H., Cohen, S., Price, B., Mai, L., Ong, S.-H., and Feng, J. (2019). Multiseg: Semantically meaningful, scale-diverse segmentations from minimal user input. In *ICCV*, pages 662–670. ix, xviii, 10, 24, 25, 26, 80, 81, 82, 85, 86, 87, 89, 90, 110, 111

Lin, D., Dai, J., Jia, J., He, K., and Sun, J. (2016). Scribblesup: Scribble-supervised convolutional networks for semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3159–3167. 3

Lin, T.-Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Dollár, P., and Zitnick, C. L. (2014). Microsoft COCO: Common objects in context. In *ECCV*. 4, 5, 14, 22, 41, 43, 47, 54, 67, 74, 75, 81, 82, 86, 89, 98, 101, 102, 110

Lin, Z., Zhang, Z., Chen, L.-Z., Cheng, M.-M., and Lu, S.-P. (2020). Interactive image segmentation with first click attention. In *CVPR*, pages 13339–13348. 5, 7, 24, 26, 42, 114

Ling, H., Gao, J., Kar, A., Chen, W., and Fidler, S. (2019). Fast interactive object annotation with curve-gcn. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5257–5266. 3, 4, 5, 29

Liu, D., Zhang, D., Song, Y., Zhang, C., Zhang, F., O'Donnell, L., and Cai, W. (2019). Nuclei segmentation via a deep panoptic model with semantic feature fusion. In *AAAI*, pages 861–868. 96

Lombaert, H., Sun, Y., Grady, L., and Xu, C. (2005). A multilevel banded graph cuts method for fast image segmentation. In *ICCV*, volume 1, pages 259–265. IEEE. 20, 112

Long, J., Shelhamer, E., and Darrell, T. (2015). Fully convolutional networks for semantic segmentation. In *CVPR*. x, xvii, 2, 3, 4, 6, 14, 21, 22, 31, 36, 37, 42, 48, 54, 55, 59, 66, 68, 76, 81, 82, 83, 84, 87, 95, 96, 98, 101, 108, 110

Maas, A. L., Hannun, A. Y., Ng, A. Y., et al. (2013). Rectifier nonlinearities improve neural network acoustic models. In *ICML*, volume 30, page 3. Citeseer. 33, 98

Mahadevan, S., Voigtlaender, P., and Leibe, B. (2018). Iteratively trained interactive segmentation. In *BMVC*. xiv, xvii, xviii, xix, 4, 5, 6, 7, 8, 9, 10, 14, 24, 27, 42, 46, 47, 48, 49, 52, 53, 54, 55, 59, 66, 68, 69, 70, 72, 74, 77, 78, 80, 81, 82, 85, 86, 87, 88, 89, 90, 91, 94, 95, 96, 98, 100, 101, 102, 103, 104, 105, 106, 107, 108, 109, 110, 112

Majumder, S., Khurana, A., Rai, A., and Yao, A. (2020a). Multi-stage fusion for one-click segmentation. In *Pattern Recognition*, pages 174–187, Cham. Springer International Publishing. 8, 12, 110

Majumder, S., Rai, A., Khurana, A., and Yao, A. (2020b). Two-in-one refinement for interactive segmentation. In *Proc. 31st British Machine Vision Conference (BMVC20)*. 8, 12, 110, 112

Majumder, S. and Yao, A. (2019a). Content-aware multi-level guidance for interactive instance segmentation. In *CVPR*, pages 11602–11611. xiv, xvii, xviii, xix, 7, 8, 10, 12, 65, 66, 67, 69, 70, 71, 74, 75, 76, 77, 81, 82, 83, 86, 87, 88, 89, 90, 91, 96, 100, 101, 103, 104, 105, 108, 109, 111

Majumder, S. and Yao, A. (2019b). Localized interactive instance segmentation. In Fink, G. A., Frintrop, S., and Jiang, X., editors, *Pattern Recognition*, pages 522–536, Cham. Springer International Publishing. xviii, 8, 12, 77, 82, 89, 90, 96, 100, 108

Malcolm, J., Rathi, Y., and Tannenbaum, A. (2007). Graph cut segmentation with nonlinear shape priors. In *ICIP*, volume 4, pages IV–365. IEEE. 19

Maninis, K.-K., Caelles, S., Pont-Tuset, J., and Van Gool, L. (2018). Deep extreme cut: From extreme points to object segmentation. In *CVPR*. viii, xiv, 1, 4, 7, 9, 10, 14, 15, 24, 27, 46, 48, 52, 65, 66, 68, 69, 71, 74, 75, 77, 79, 81, 82, 83, 89, 91, 108, 110, 111

Maninis, K.-K., Pont-Tuset, J., Arbeláez, P., and Van Gool, L. (2016). Convolutional oriented boundaries. In *ECCV*. 9, 52, 108

Martin, D. R., Fowlkes, C. C., and Malik, J. (2004). Learning to detect natural image boundaries using local brightness, color, and texture cues. *IEEE PAMI*, 26(5):530–549. 41

McGuinness, K. and O'connor, N. E. (2010). A comparative evaluation of interactive segmentation algorithms. *Pattern Recognition*, 43(2):434–444. xvii, 41, 43, 47, 54, 74, 75, 76, 87, 89, 101

McInerney, T. and Terzopoulos, D. (2000). T-snakes: Topology adaptive snakes. *Medical image analysis*, 4(2):73–91. 13, 15

Menet, S., Saint-Marc, P., and Medioni, G. (1990). Active contour models: overview, implementation and applications. In *1990 IEEE International Conference on Systems, Man, and Cybernetics Conference Proceedings*, pages 194–199. 15

Mortensen, E. N. and Barrett, W. A. (1995). Intelligent scissors for image composition. In *SIGGRAPH*. viii, 2, 5, 13, 15, 16, 17, 18, 20, 26, 48, 67, 82, 95

Mortensen, E. N. and Barrett, W. A. (1998). Interactive segmentation with intelligent scissors. *Graphical models and image processing*, 60(5):349–384. viii, 16

Niemeijer, M., Staal, J., van Ginneken, B., Loog, M., and Abramoff, M. D. (2004). Comparative study of retinal vessel segmentation methods on a new publicly available database. In *Medical imaging 2004: image processing*, volume 5370, pages 648–656. International Society for Optics and Photonics. vii, 2

Noh, H., Hong, S., and Han, B. (2015). Learning deconvolution network for semantic segmentation. In *ICCV*. vii, 5, 6, 9, 47, 56, 58, 60, 111

Oh, S. W., Lee, J.-Y., Xu, N., and Kim, S. J. (2019). Fast user-guided video object segmentation by interaction-and-propagation networks. In *CVPR*, pages 5247–5256. 27, 113, 114

Papadopoulos, D. P., Uijlings, J. R., Keller, F., and Ferrari, V. (2017). Extreme clicking for efficient object annotation. In *ICCV*, pages 4930–4939. 110

Papandreou, G., Kokkinos, I., and Savalle, P.-A. (2015). Modeling local and global deformations in deep learning: Epitomic convolution, multiple instance learning, and sliding window detection. In *CVPR*. 6, 47, 52

Papazoglou, A. and Ferrari, V. (2013). Fast object segmentation in unconstrained video. In *ICCV*. 50, 70

Park, T., Liu, M.-Y., Wang, T.-C., and Zhu, J.-Y. (2019). Semantic image synthesis with spatially-adaptive normalization. In *CVPR*, pages 2337–2346. 95, 97

Perazzi, F., Pont-Tuset, J., McWilliams, B., Van Gool, L., Gross, M., and Sorkine-Hornung, A. (2016). A benchmark dataset and evaluation methodology for video object segmentation. In *CVPR*. 74, 75, 78, 87, 89

Pont-Tuset, J., Arbelaez, P., Barron, J. T., Marques, F., and Malik, J. (2017). Multiscale combinatorial grouping for image segmentation and object proposal generation. *TPAMI*, 39(1):128–140. xi, 9, 39, 40, 41, 46, 52, 55, 58, 71, 108, 109

Prewitt, J. M. and Mendelsohn, M. L. (1966). The analysis of cell images. *Annals of the New York Academy of Sciences*, 128(3):1035–1053. 1

Price, B. L., Morse, B., and Cohen, S. (2010). Geodesic graph cut for interactive image segmentation. In *CVPR*. 13, 18, 21, 48, 68

Rakelly, K., Shelhamer, E., Darrell, T., Efros, A. A., and Levine, S. (2018). Few-shot segmentation propagation with guided networks. *arXiv preprint arXiv:1806.07373*. xv, 11, 95, 96, 97, 102, 109

Rantalankila, P., Kannala, J., and Rahtu, E. (2014). Generating object segmentation proposals using global and local search. In *CVPR*, pages 2417–2424. 39

Ren, X. and Malik, J. (2003). Learning a classification model for segmentation. In *ICCV*, volume 2, pages 10–10. IEEE Computer Society. 39

Rosenfeld, A. (1976). *Digital picture processing*. Academic press. 1

Rother, C., Kolmogorov, V., and Blake, A. (2004). Grabcut: Interactive foreground extraction using iterated graph cuts. *ACM transactions on graphics (TOG)*, 23(3):309–314. viii, 1, 2, 5, 13, 15, 17, 18, 20, 26, 41, 43, 46, 47, 48, 54, 65, 67, 74, 75, 78, 87, 88, 90, 96, 101, 112, 114

Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., et al. (2015). Imagenet large scale visual recognition challenge. *IJCV*, 115(3):211–252. x, 13, 14, 21, 31, 32, 34, 74, 98, 100

Shi, J., Yan, Q., Xu, L., and Jia, J. (2015). Hierarchical image saliency detection on extended cssd. *IEEE TPAMI*, 38(4):717–729. 96, 101, 102

Shu, G., Dehghan, A., and Shah, M. (2013). Improving an object detector and extracting regions using superpixels. In *CVPR*, pages 3721–3727. 39

Simonyan, K. and Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*. ix, x, xv, 2, 14, 25, 31, 34, 35, 36, 37, 59, 95, 97

Singh, B. and Davis, L. S. (2018). An analysis of scale invariance in object detection–snip. In *CVPR*. xii, 52, 56, 57

Sinop, A. K. and Grady, L. (2007). A seeded image segmentation framework unifying graph cuts and random walker which yields a new algorithm. In *ICCV*, pages 1–8. IEEE. 21

Slabaugh, G. and Unal, G. (2005). Graph cuts segmentation using an elliptical shape prior. In *IEEE International Conference on Image Processing 2005*, volume 2, pages II–1222. IEEE. 19

Sofiiuk, K., Petrov, I., Barinova, O., and Konushin, A. (2020). F-brs: Rethinking backpropagating refinement for interactive segmentation. In *CVPR*. xiv, 5, 7, 25, 26, 80, 81, 82, 94, 98, 100, 103, 105, 107, 108, 109, 110, 111, 112

Staib, L. H. and Duncan, J. S. (1992). Boundary finding with parametrically deformable models. *PAMI*, 14(11):1061–1075. 15

Stalling, D. and Hege, H.-C. (1996). Intelligent scissors for medical image segmentation. *Digitale Bildverarbeitung fuer die Medizin*, pages 32–36. 1, 16

Stutz, D., Hermans, A., and Leibe, B. (2018). Superpixels: An evaluation of the state-of-the-art. *Computer Vision and Image Understanding*, 166:1–27. 39

Su, H., Deng, J., and Fei-Fei, L. (2012). Crowdsourcing annotations for visual object detection. In *Workshops at the Twenty-Sixth AAAI Conference on Artificial Intelligence*. 110

Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., and Rabinovich, A. (2015). Going deeper with convolutions. In *CVPR*, pages 1–9. 14

Uijlings, J. R., Van De Sande, K. E., Gevers, T., and Smeulders, A. W. (2013). Selective search for object recognition. *IJCV*, 104(2):154–171. 52, 55

Van den Bergh, M., Boix, X., Roig, G., de Capitani, B., and Van Gool, L. (2012). Seeds: Superpixels extracted via energy-driven sampling. In *ECCV*, pages 13–26. Springer. xi, 39, 40, 46

Vedaldi, A. and Lenc, K. (2015). Matconvnet: Convolutional neural networks for matlab. In *MM*. 54

Veksler, O. (2002). Stereo correspondence with compact windows via minimum ratio cycle. *PAMI*, 24(12):1654–1660. 19

Veksler, O. (2008). Star shape prior for graph-cut image segmentation. In *ECCV*. 19

Vezhnevets, V. and Konouchine, V. (2005). Growcut: Interactive multi-label nd image segmentation by cellular automata. In *Graphicon*. 13, 48, 67

Vielzeuf, V., Pateux, S., and Jurie, F. (2017). Temporal multimodal fusion for video emotion classification in the wild. In *Proceedings of the 19th ACM International Conference on Multimodal Interaction*, pages 569–576. ACM. 96

Vincent, L. and Soille, P. (1991). Watersheds in digital spaces: an efficient algorithm based on immersion simulations. *IEEE Computer Architecture Letters*, 13(06):583–598. 4, 20, 46

Vu, N. and Manjunath, B. (2008). Shape prior segmentation of multiple objects with graph cuts. In *CVPR*, pages 1–8. IEEE. ix, 19

Wacker, A. C. and Landgrebe, D. A. (1970). Boundaries in multispectral imagery by clustering. In *1970 IEEE Symposium on Adaptive Processes (9th) Decision and Control*, pages 114–114. IEEE. 1

Wan, J., Wang, D., Hoi, S. C. H., Wu, P., Zhu, J., Zhang, Y., and Li, J. (2014). Deep learning for content-based image retrieval: A comprehensive study. In *Proceedings of the 22nd ACM international conference on Multimedia*, pages 157–166. 1

Wang, G., Li, W., Zuluaga, M. A., Pratt, R., Patel, P. A., Aertsen, M., Doel, T., David, A. L., Deprest, J., Ourselin, S., et al. (2018a). Interactive medical image segmentation using deep learning with image-specific fine-tuning. *IEEE Transactions on Medical Imaging*. 5, 13, 46, 65, 79

Wang, G., Zuluaga, M. A., Li, W., Pratt, R., Patel, P. A., Aertsen, M., Doel, T., Divid, A. L., Deprest, J., Ourselin, S., et al. (2018b). DeepIGeoS: a deep interactive geodesic framework for medical image segmentation. *TPAMI*. 1, 5, 46, 65

Wang, X., Kong, T., Shen, C., Jiang, Y., and Li, L. (2019). Solo: Segmenting objects by locations. *arXiv preprint arXiv:1912.04488*. 2

Weszka, J. S. (1978). A survey of threshold selection techniques. *Computer graphics and Image processing*, 7(2):259–265. 1

Xiaofeng, R. and Bo, L. (2012). Discriminatively trained sparse code gradients for contour detection. *NIPS*, 25. 41

Xiong, Y., Liao, R., Zhao, H., Hu, R., Bai, M., Yumer, E., and Urtasun, R. (2019). Upsnet: A unified panoptic segmentation network. In *CVPR*, pages 8818–8826. 3

Xu, N., Price, B., Cohen, S., and Huang, T. (2017a). Deep image matting. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2970–2979. 113

Xu, N., Price, B., Cohen, S., Yang, J., and Huang, T. (2017b). Deep grabcut for object selection. In *BMVC*. viii, 7, 10, 15, 22, 23, 66, 67, 68, 108, 112

Xu, N., Price, B., Cohen, S., Yang, J., and Huang, T. S. (2016). Deep interactive object selection. In *CVPR*. vii, viii, ix, xii, xiv, xvii, xviii, xix, 2, 4, 6, 7, 9, 10, 13, 14, 15, 21, 22, 23, 24, 27, 36, 42, 43, 45, 46, 47, 48, 49, 51, 52, 53, 54, 55, 56, 58, 59, 65, 66, 68, 69, 70, 72, 74, 75, 76, 77, 78, 80, 81, 82, 83, 84, 85, 86, 87, 88, 89, 90, 91, 94, 95, 96, 98, 99, 100, 101, 103, 104, 105, 106, 107, 108, 109, 110, 112

Xue, Y., Tang, H., Qiao, Z., Gong, G., Yin, Y., Qian, Z., Huang, C., Fan, W., and Huang, X. (2020). Shape-aware organ segmentation by predicting signed distance maps. In *AAAI*, volume 34, pages 12565–12572. 84

Yan, J., Yu, Y., Zhu, X., Lei, Z., and Li, S. Z. (2015). Object detection by labeling superpixels. In *CVPR*, pages 5107–5116. 39

Yao, J., Boben, M., Fidler, S., and Urtasun, R. (2015). Real-time coarse-to-fine topologically preserving segmentation. In *CVPR*. xi, 9, 39, 40, 46, 47, 55, 58, 108

Ye, Q.-Z. (1988). The signed Euclidean distance transform and its applications. In *ICPR*. 84

Yu, C., Wang, J., Peng, C., Gao, C., Yu, G., and Sang, N. (2018). Learning a discriminative feature network for semantic segmentation. In *CVPR*, pages 1857–1866. 67

Yu, F. and Koltun, V. (2016). Multi-scale context aggregation by dilated convolutions. In *ICLR*. 59

Yu, Z., Feng, C., Liu, M.-Y., and Ramalingam, S. (2017). Casenet: Deep category-aware semantic edge detection. In *CVPR*. 82

Zeiler, M. D. and Fergus, R. (2014). Visualizing and understanding convolutional networks. In *ECCV*, pages 818–833. Springer. 2

Zeiler, M. D., Taylor, G. W., and Fergus, R. (2011). Adaptive deconvolutional networks for mid and high level feature learning. In *ICCV*, pages 2018–2025. IEEE. 31

Zhang, H., Dana, K., Shi, J., Zhang, Z., Wang, X., Tyagi, A., and Agrawal, A. (2018). Context encoding for semantic segmentation. In *CVPR*, pages 7151–7160. 67

Zhang, Y., Gong, L., Fan, L., Ren, P., Huang, Q., Bao, H., and Xu, W. (2019). A late fusion cnn for digital matting. In *CVPR*, pages 7469–7478. 96

Zhao, H., Shi, J., Qi, X., Wang, X., and Jia, J. (2017). Pyramid scene parsing network. In *CVPR*, pages 2881–2890. 2, 74, 83, 98, 99