Imperial College of Science, Technology and Medicine
Department of Computing

Exploring Neural Representations for Self-Supervised Segmentation

Anagh Malik

In part fulfilment of the requirements for the degree of
Master of Research in Artificial Intelligence and Machine Learning
the Diploma of Imperial College
August 2022

Abstract

Image and scene segmentation is one of the central tasks of Computer Vision. A segmentation could be used as an efficient representation for an early vision system, which could then be applied to various downstream tasks. A good segmentation map might be useful for applications in autonomous driving or robotic manipulation.

Most segmentation algorithms learn useful features from large labelled datasets. However in many domains such as medical imaging or astrophysics ground truth segmentations might be unavailable. This is where self-supervised segmentation comes in, where we try to learn features which lend themselves well to segmentation, without any labelled datasets.

Neural Representations are an emerging method to represent signals, by parameterizing them with a neural network. This representation lends itself well to the task of self-supervised segmentation, due to the compressive and hierarchical nature of neural networks.

In this thesis we explore the use of *Neural Representations* for efficient self-supervised segmentation without any pretraining. We first introduce *SegDIP*, a method for the well-defined problem of interactive segmentation using a Convolutional Neural Field. We develop different kinds of user interactions, which make the process of interactive segmentation simpler. We analyze the properties of the network using the Neural Tangent Kernel literature. Finally we show some applications of our method for conditional image generation and semi-supervised video tracking.

We then introduce a method for self-supervised segmentation, making use of *Neural Representations* to discover semantically meaningful contours, which are then used for producing a segmentation mask. Our method, inspired by the work done in SegDIP, uses a multi-head self distillation setup to ensure agreement between multiple segmentation hypotheses, which then help guide our results for the final contours.

Acknowledgements

I am particularly grateful to my supervisor Prof. Andrew Davison for providing me with the opportunity to join the Dyson Robotics Lab. Through his intuitive ideas and insightful questions he has been able to steer all my projects into meaningful scientific contributions. His emphasis on depth of thought is one thing I will especially not forget.

I am equally grateful to Dr. Ronald Clark, who also agreed to support me as a supervisor during my studies. Through his mastery of current topics in the field, he was always able to guide me to make more impactful contributions with my work. His emphasis on placing ones work amongst current literature is something I will keep in mind.

I feel very fortunate being part of the Dyson Robotics Lab, where I had the opportunity to collaborate with many excellent researchers including Shuaifeng Zhi, Edgar Sucar and Marwan Taher.

I am also thankful to all other members of the Dyson Robotics Lab and its affiliates, all of whom I would consider my friends: Kentaro Wada, Tristan Laidlow, Daniel Lenton, Joe Ortiz, Eric Dexheimer, Dorian Henning, Ivan Kapelyukh, Xin Kong, Zoe Landgraf, Shikun Liu, Hidenobu Matsuki, Kirill Mazur, Riku Murai, Seth Nabarro, Iain Haughton, Andre Mouton and Aalok Patwardhan.

Last but not least, I would like to thank my parents and my brother for their unconditional support and care throughout my time at Imperial College London.

Contents

Chapter 1

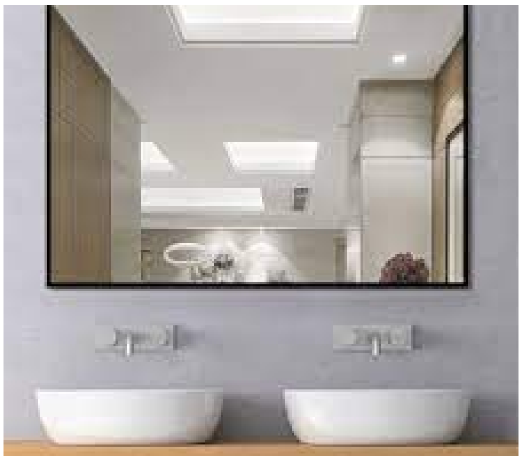
Background & Introduction

It is surprising how much information a human can deduce about a scene from just a glance of it. We can recognize compositional information, estimate material properties or develop partial understanding of the occluded objects. One of the central aspects to our scene understanding is the ability to separate objects, for example I can perfectly point out where the paper in front of me ends and the desk starts.

This problem in Computer Vision is called *segmentation*. The definition of image segmentation is as simple as it gets - image segmentation is the task of assigning a discrete label to each pixel in the image. This assignment is meant to separate an image into regions, which should be representing physical scene properties (like objects or object parts), useful for any other task down the line.

However there is a fundamental problem in this task definition, we do not know beforehand what the labels are. If we have an image of a human, should his fingers be separate objects? Should his nose be a separate object? Are they perhaps just a part of the bigger object of the human? As David Marr says in his influential book on vision [Mar82], it is well neigh impossible to formulate precisely what the exact goals of segmentation are.

However even assuming we manage to fix this definition issue, there are even more intricacies to the problem than we realize. The global understanding required for segmentation is


(a) Recognition issue. How do we know the mirror is not a window? It comes from our analysis of the whole image, by deducting the scene is set in a bathroom.

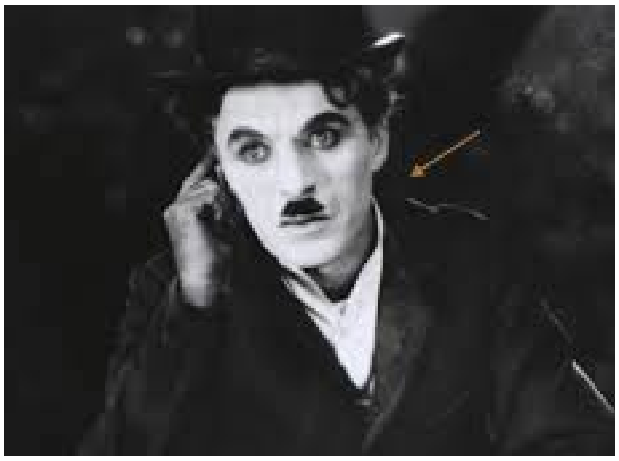

(b) Segmentation issue. How do we tell where the coat ends and background begins (red arrow)? Even humans can not do it exactly, but we can probably do pretty good job just by using the idea of continuity in the shape of the coat and analyzing where the shirt ends.

Figure 1.1: Images difficult for semantic segmentation.

unparalleled. The problem becomes even more difficult when you realize that simply seeing the object you have at hand is oftentimes not enough. Many times you have to also perceive and understand the surroundings. For example looking at Figure 1.1a, if I have an image of a bathroom how do I know a mirror on the wall is not a window? Or in Figure 1.1b, how do I tell where the background begins and the coat ends?

The hierarchical scene understanding humans have developed, which allows us to recognize and separate objects visually in an open set manner is unparalleled. Nonetheless many brave Computer Vision researchers have attempted to solve this problem or at least contribute to solving it.

Before we delve into some of the work done in this field, it is important to mention a slight nuance in the problem definition. The label set might not explicitly represent any objects a priori, in this case we are dealing with the problem of *segmentation*. However we might also be simultaneously solving the problem of segmentation and object recognition i.e. when the labels have objects associated with them a priori. This problem is called *semantic segmentation*.

1.1 Early days of segmentation

Edge detection. Prior to tackling the problem of segmentation, many researchers concerned themselves with edge detection. The idea behind edge detection is to find sharp changes in an image. The hope is that these changes in the image are caused by changes in scene properties like depth or material, which then correspond to object boundaries.[1]

Leo Hodes in 1961 [Hod61] did some of the earliest work on detecting edges in line drawings.[2] However Lawrence Roberts in 1963 did most influential early work in this space [Rob63]. He introduced what is known at the Roberts cross operator. Given a grayscale image, which we define by its pixels $x_{ij} \in \mathbb{R}_{\geq 0}$, we define the image derivatives as $z_{ij} \in \mathbb{R}_{\geq 0}$:

$$y_{i,j} = \sqrt{x_{i,j}} \tag{1.1}$$

$$z_{i,j} = \sqrt{\left(y_{i,j} - y_{i+1,j+1}\right)^2 + \left(y_{i+1,j} - y_{i,j+1}\right)^2} \tag{1.2}$$

More commonly we describe the filtering operation as magnitude of the vector resulting from application (convolution with) of the two kernels:

$$\begin{bmatrix} +1 & 0 \\ 0 & -1 \end{bmatrix} \text{ and } \begin{bmatrix} 0 & +1 \\ -1 & 0 \end{bmatrix}$$

After recognizing points with a high gradient value, Roberts in his work intended to connect these dots with lines. 3D models were then fitted to these line drawings, to do tasks such as novel view synthesis (at least a novel view of the edges). We can see that the filter is mainly equipped to recognize diagonal changes in the image.

Early work on edges continued and in 1968 during a talk Irwin Sobel introduced the Sobel filter [Sob14]. It is calculated as the magnitude of the vector after applying the following two

[1]These days object boundaries are more commonly referred to as contours and edges are just supposed to indicate local changes in images.

[2]It is not easy to find the original source. Roberts refers to the work in his thesis.

kernels:

$$\begin{bmatrix} +1 & 0 & -1 \\ +2 & 0 & -2 \\ +1 & 0 & -1 \end{bmatrix} \quad \text{and} \quad \begin{bmatrix} +1 & +2 & +1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}$$

The filter is less prone to noise, but more expensive to use.

We would not be able to end our discussion on classical edge detectors without mentioning Canny [Can86], introduced by John Canny in 1986. Canny starts off its edge detection by first smoothing the image with a Gaussian filter. This is to remove noise from the image and to avoid detecting very high frequency patterns. Then a filtering operation is applied (like Roberts or Sobel), to find the edges. After that, Canny performs non-maximal suppression, which is an operation that calculates gradient directions to thin out the edges (along them). The next step is to bin all the edges into 3 categories, "edge", "possible edge" and "not edge", using pre-determined thresholds on the gradient maps. Finally hysteresis is performed, the algorithm goes along "sure" edges and if it crosses a "possible edge" during its path, then those pixels are considered "sure" edges as well. Canny is considered one of the most important edge detectors, sill widely used today.

The results from all three edge detectors can be seen in Figure 1.2. The usefulness of edges or contours comes from them being label agnostic i.e. if we have multiple edge representations they already live in the same domain, for example 0 represents no edge, 1 represents edge etc. This is as opposed to segmentation, which have an unknown domain (number of labels, if semantic then numbering of those classes). It is important to keep in mind that it is not obvious how to represent semantic classed with edges.

Edge detection methods will make a return later in our work and overview (disguised as contours), however we might want to return to the problem of direct region detection.

Direct region detection. We have discussed some early edge detection work, however this is of course not a necessary step to go to segmentation. In his 1968 thesis [Guz68], Adolfo Guzman found a way to use vertices in images to isolate out separate objects. This work

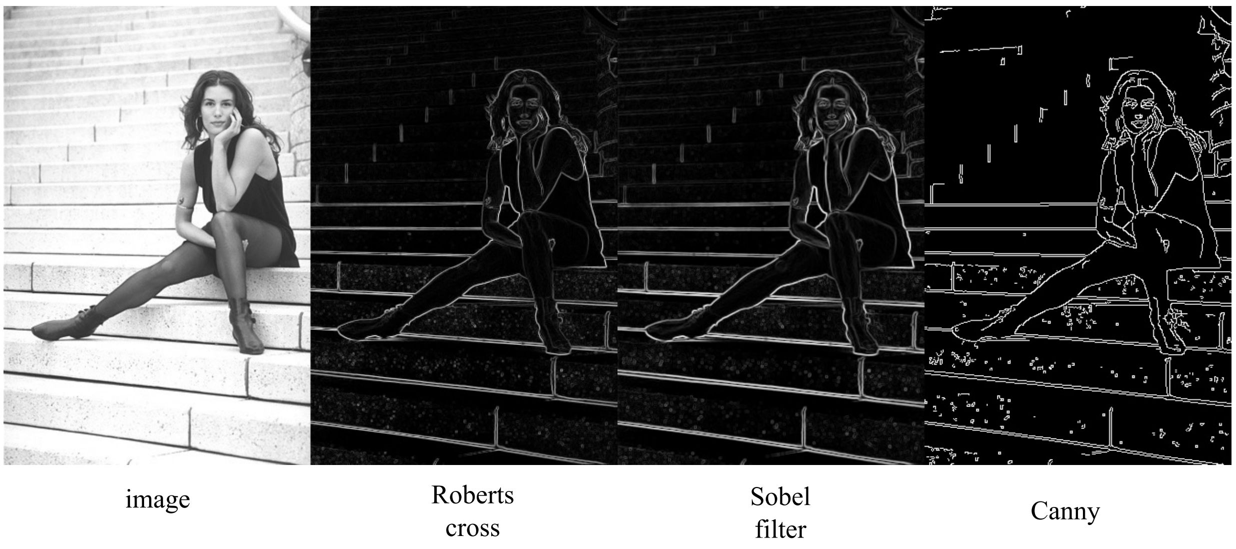


Figure 1.2: Results of operation of different edge detection methods.

marked a slight change towards a trend of recognizing multiple regions from images.

It was only in 1971 [MP71] that Minsky and Papert introduced the first method for segmentation on grayscale images without an intermediate representations of edges or vertices. Their method relied on finding regions, which were constructed of unions of squares whose corners have similar grayscale values.

Many other methods were developed in the 1970s. The 1981 review [FM81] points out the two leading trends. The first one was characteristic feature thresholding and clustering based segmentation [WNR74], [HD75], [SDR76]. These methods intend on first describing pixels through some features i.e. aggregating local brightness, texture cues etc. Then by simple thresholding on these features they are able to bin the pixels into separate classes. The second major trend the review paper describes are region merging and splitting [JMB76], [GW74a], [GW74b]. These are iterative algorithms often known as "region-growing" algorithms, which refine their region predictions.

Up until this point the segmentation methods were largely "local", and analyzed local image cues to group regions together, without defining or optimizing any global objectives.

In 1980s the dominating techniques relied on Markov Random Fields (MRF) [GG84], which are able to rely on global objectives of the image as opposed to local feature thresholding. In short for MRF based techniques, you define a probability measure on the labelling space which follows the Gibbs distribution. Given *features for each pixel*, you can then solve the labelling by MAP estimate. Such an optimization framework setup leads to more "global" understanding of the image due to solving a simultaneous objective for the whole image.

1.2 Classical methods for segmentation

With Computer Vision becoming more popular[3], the late 1990s and early 2000s saw a major boom in segmentation algorithms. We will not provide a comprehensive review here of those, but we will cover some of the more important ones, which will also serve as benchmarks for some of the work covered in this thesis.

Normalized Cuts. Normalized Cuts were introduced by Shi and Malik in 1998 [SM00] [4]. In normalized cuts we treat the segmentation problem as a graph cut problem. We are given a fully-connected weighted graph, where the nodes represent the pixels and weights $w(a, b)$ represent the similarity between any two pixels a and b. Given two groups of pixels A and B, we can define the cost of a cut (separating them as classes) between them as:

$$\text{cut}(A, B) = \sum_{u \in A, v \in B} w(u, v)$$

Then given n classes the objective is to cut the graph (along the edges) in such a way as to minimize the cost of all cuts (sum of weights between each pixel in any two separate classes). Of course a cheaper cut overall is to separate the image into some very small groups, this is where the word *normalized* comes in. We can calculate a normalization constant as:

[3]The 70s and 80s are considered AI winter years. This is definitely visible in the number of breakthrough methods during these times as compared to late 90s and early 2000s.

[4]This citation is their follow up journal entry from 2000, where specifically build on the older work.

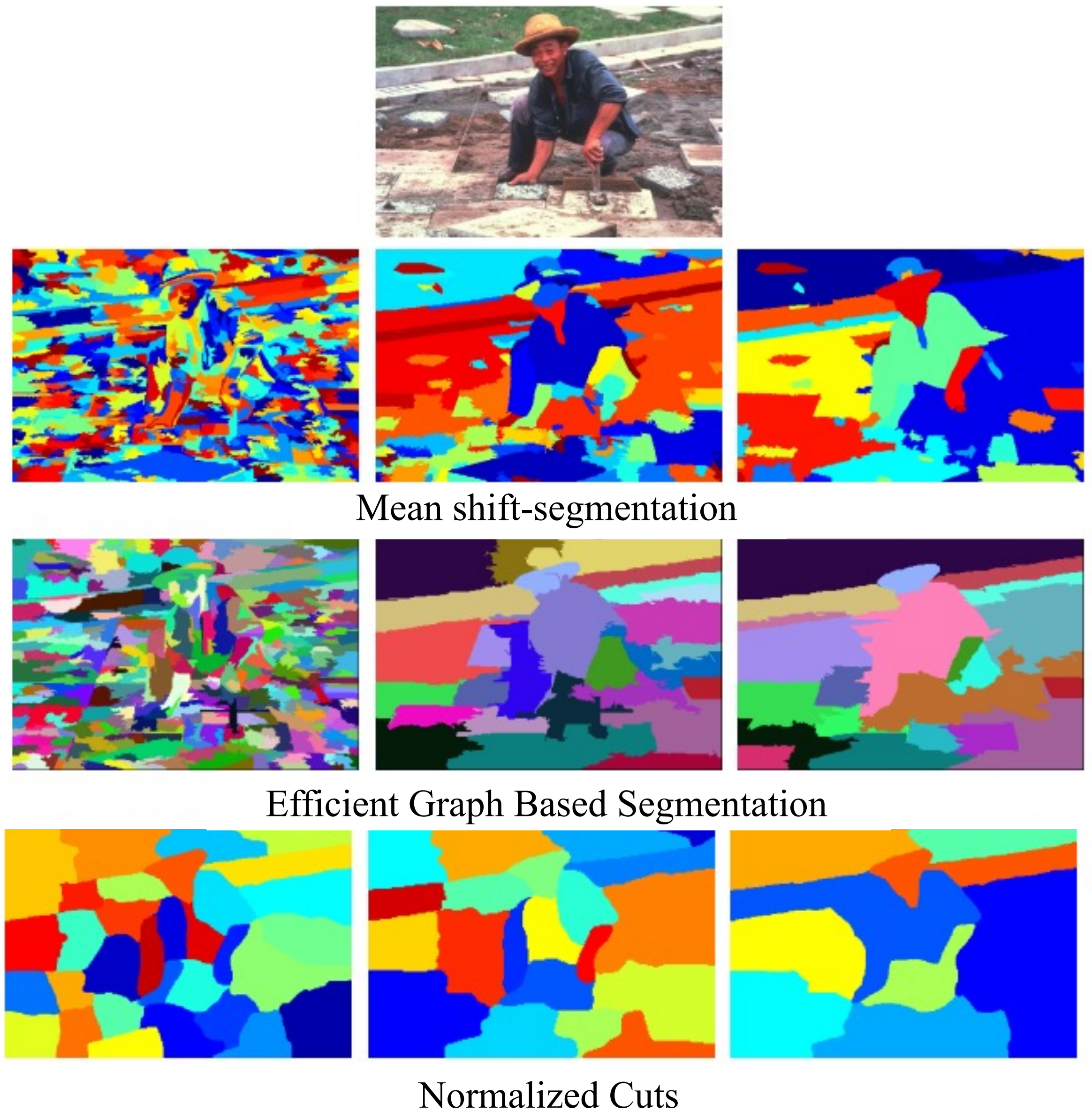


Figure 1.3: Results of operation of different unsupervised segmentation methods. Each method is run with three separate parameter settings. Figure redacted from [Pan08].

$$\mathrm{assoc}(A, V) = \sum_{u \in A, t \in V} w(u, t)$$

where V is the whole image. Then we can calculate the normalized cost as:

$$\mathrm{Ncut}(A, B) = \frac{\mathrm{cut}(A, B)}{\mathrm{assoc}(A, V)} + \frac{\mathrm{cut}(A, B)}{\mathrm{assoc}(B, V)}$$

The objective now would be normalized and hence not biased to smaller classes. It would however be biased to equal sized classes, hence all classes would get more or less equal sized chunks.

The way the cut is calculated (objective minimzed) is by framing the problem as a linear algebra one, the problem can be posed as an eigenvector solving problem.

The algorithm is really quite flexible, since the weights can be defined as any sort of affinity. In the original paper for grayscale images a combination (exponential for affinity) distance of pixel intensities and coordinate distance is used. These ideas have been extended to soft segments by spectral matting using a matting affinity matrix [LRAL07] and to semantic spectral matting using features from a network trained for semantic segmentation [AOP$^+$18].

There is an obvious issue in the formulation of the graph problem, it is highly inefficient. Calculating this huge affinity matrix is not very feasible and follow up work cited, recognizes this and uses superpixels as chunks to find affinities for. However we then rely on the shortcomings of superpixels. Furthermore it would not be obvious how to generalize these ideas to the 3D domain, given that we would have to calculate an affinity between each and every point.

There are many connections to be made between normalized cuts and the method we will present in the second chapter, however the reader will have to wait for those!

Efficient Graph-Based Segmentation. In the 2004 paper [FH04] Felzenszwalb and Huttenlocher defined an efficient graph based segmentation method. Let again the vertices of a fully connected weighted graph be pixels of an image (in the notation below the edge strength is the weightage). Their algorithm begins with setting the each pixel as a separate class. Then an objective is defined to sequentially combine the classes into a final segmentation [Pan08]:

1. Sort $E = (e_1, \ldots, e_m)$ such that $|e_t| \leq |e_{t'}| \, \forall t < t'$
2. Let $S^0 = (\{\mathbf{x}_1\}, \ldots, \{\mathbf{x}_n\})$, in other words each initial cluster contains exactly one vertex.
3. For $t = 1, \ldots, m$
   - Let $\mathbf{x}_i$ and $\mathbf{x}_j$ be the vertices connected by e_t.
   - Let $C^{t-1}_{\mathbf{x}_i}$ be the connected component containing point $\mathbf{x}_i$ on iteration $t-1$, and $l_i = \max_{\text{mst}} C^{t-1}_{\mathbf{x}_i}$ be the longest edge in the minimum spanning tree of $C^{t-1}_{\mathbf{x}_i}$. Likewise for l_j.
   - Merge $C^{t-1}_{\mathbf{x}_i}$ and $C^{t-1}_{\mathbf{x}_j}$ if

     $$|e_t| < \min\left\{ l_i + \frac{k}{\left|C^{t-1}_{\mathbf{x}_i}\right|}, l_j + \frac{k}{\left|C^{t-1}_{\mathbf{x}_j}\right|} \right\}$$

     where k is a constant.
4. $S = S^m$

The paper also proposes simplification of the graph, by defining edges only for the nearest neighbours in metric of the feature space. The features considered for the distances were again just colour (intensity) and coordinate distances.

Mean Shift Segmentation. The mean shift for image segmentation was first introduced in 2002 by Comaniciu and Meer [CM02]. The simple idea behind the algorithm is that you should be able to find modes of the pixel feature distributions. Then pixels, which have features near these nodes can be clustered with them, segmenting the image.

The specifics on the mean-shift algorithm define a way to find these modes of the distribution. Initially pixels were represented by their colour in the $L^*u^*v^*$ colour space and their location (x, y). Then, in short, an iterative algorithm is applied to recompute the features describing each pixel, until these features converge to a few modes.

Feature learning. Results of the segmentation methods discussed above can be seen in Figure 1.3. There is a trend to be seen in all of these methods. The features they use to define a pixel is usually not very sophisticated and just based on colour (intensity) and coordinate values. The works are not about *feature learning*, but rather defining algorithms to find a clustering on the features. They are quite agnostic to whatever features one defines. There will be a shift we will see, where the feature definitions will get more complicated in the future and even farther (and in our work), feature learning will become the main goal for many.

1.3 Supervised Segmentation

Labelled datasets and methods leveraging them. We could consider the late 2000s and 2010s the advent of "modern" Computer Vision. The field evolved to put a higher emphasis on labelled datasets. An example of this is ImageNet [DDS+09], a dataset with 14 million images, with class labels. ImageNet also gave birth to the ImageNet Large Scale Visual Recognition Challenge (ILSVRC). Datsets for segmentation and edge detection were also introduced, like the Berkeley Segmentation Dataset (BSDS300) [MFTM01], with initially just 300 images for segmentation with groundtruth labels and edges.

This brought about many methods for segmentation and edge detection. In particular for both edge detection and segmentation Pb [MFM04] and gPb [AMFM11], which used the BSDS300 to learn an edge detector. This time more complicated methods are used to define the features for pixels, these are for example brightness, colour gradients and textons. Textons introduced in 2001 [MBLS01] uncover an edge based feature for each pixel. Over a dataset

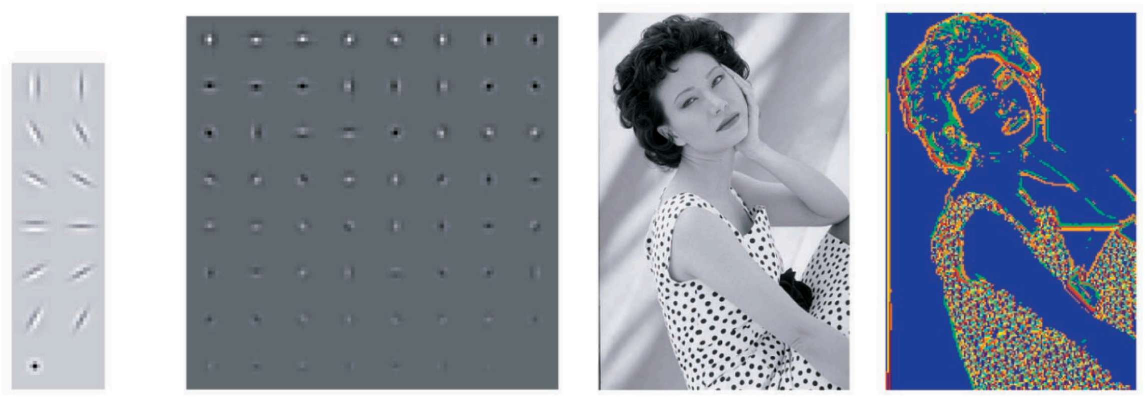


Figure 1.4: Original filters, textons deduced from a dataset, image and texton responses. Figure redacted from [MFM04].

of many images the pixels are first represented by their the response to various edge filters (orientation), then a k-means step is performed to find cluster centers called textons. The pixel is then attributed its nearest texton, an illustrative example is shown in Figure 1.4. A logistic regression model is then learnt over these feature responses and an automatic algorithm is applied to transform the edge maps into segmentations.

A similar idea is used in TextonBoost [SWRC06], which relied on conditional random fields (CRFs). Again it uses textons to describe image pixels and learns a semantic segmentation model over them using CRFs.

Deep Learning. A transformational moment for the field of Computer Vision is deemed the "AlexNet moment", when in 2012 researchers from University of Toronto won the ImageNet challenge using a Deep Neural Network (DNN), while leveraging the computational powers of GPUs [KSH12]. Since then DNNs and more specifically Convolutional Neural Networks (CNNs) have played a very large role in Computer Vision.

Convolutional Neural Networks. CNNs were first introduced in 1998 as LeNet [LBBH98]. A convolution is a linear transformation, which is shift equivariant on the input signal. A 2D convolutional layer is arranged in 3 dimensions: width, height and depth, where the convolutional kernel slides across the width and height dimensions, while having the same number of channels as the input feature.

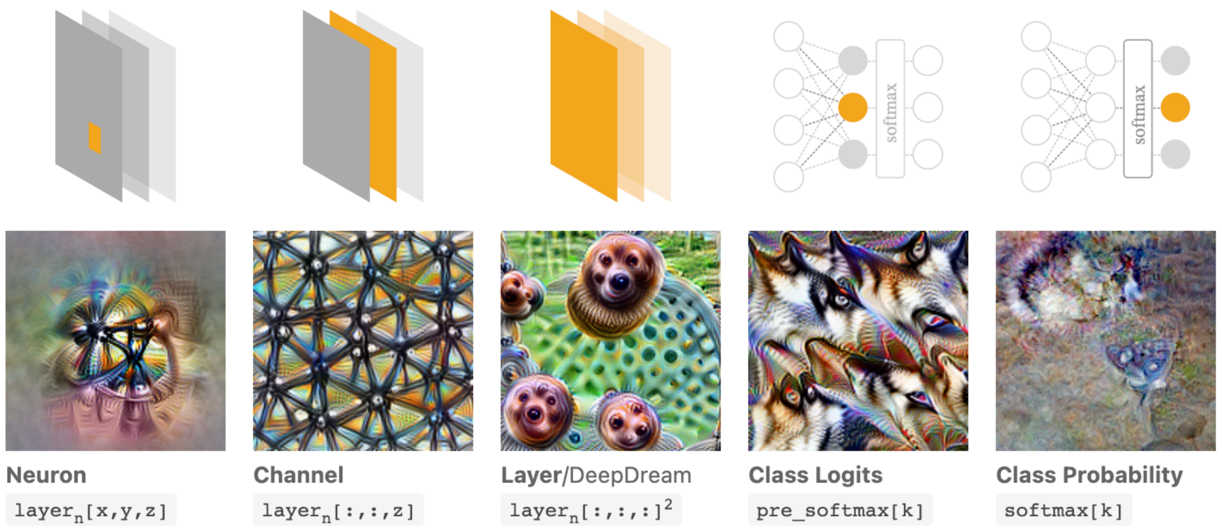


Figure 1.5: Input images which maximize the activations of the neuron(s) depicted above. Figure redacted from [OMS17].

In some tasks which require smaller spatial resolution, convolutional layers are interspersed with pooling layers, applied to reduce the dimensionality of feature maps while retaining the spatial information. This downsampling can be done through maxpooling or strided convolutions. An example of such a task would be image classification. Additionally pooling layers are also useful to increase the receptive field of a particular feature map.

CNNs can learn hierarchical object-part representations. Building on the first layer representations learned from the input images, the latter layers can learn to capture higher level information e.g. detecting ears. As seen in Figure 1.5, we can visualize this higher level information by seeing what kind of input maximizes the activation on a particular neuron of the network [OMS17].

Generative (or reconstructive) CNNs carry a similar promise, however this time earlier layers have a "higher receptive field" of the generated image, hence carrying higher level concepts of the generated image. This has however not been visualized or quantified to the best of our knowledge.

What CNNs allowed to do, which was not possible before is to learn features describing images automatically. These filters that were previously preselected as in textons, were now learnt. Moreover we could have more and higher dimensional filters to learn even higher level information. In principle, thus, the advent of CNNs allowed for replacing the hand designed features, with features learnt from the labelled datasets. An example of this is the seminal work from Long et al. [LSD15], who perform semantic segmentation using a single neural network trained end-to-end. More models with bettering performance and on different datasets followed [BKC15], [RFB15].

However, as previously mentioned, in many domains such as medical imaging or astrophysics ground truth segmentations and in general labelled datastes might be unavailable. This is where self-supervised segmentation comes in, where we try to learn features which lend themselves well to segmentation, without any labelled datasets.

1.4 Self-Supervised Learning

As previously mentioned self-supervised learning (SSL) is the learning paradigm where instead of training with ground truth data we train to optimize for some pretext tasks. This way we try to learn useful features from unlabelled datasets.

There are many pretext tasks one could solve, for example Doesrch et al. [DGE15] proposed the task of determining the ordering of patches in an image for unsupervised visual feature learning. While Pathak et al. [PAED17] proposed predicting the next frame of an episode for general pretraining of artificial agents.

Of late contrastive learning approaches have been gaining popularity [HFW+20], [CKNH20]. This is an approach where you map a patch of your image to a feature and maximize or minimize distances between features depending on whether they are generated by patches which belong to the same image.

More recently DINO [CTM+21] proposed a pretext classification task for a network to learn self-supervised features. This is also interesting since the pseudo-labels were generated by a Mean Teach [TV17]. This is a form of self-distillation, where the network's targets are produced by itself. This will be an important concept in the rest of this thesis.

Features from DINO [CTM+21] have been used for self-supervised segmentation. In most these meethods feature learning is combined with older clustering methods. For example STEGO [HZH+22] proposes to distill DINO features into smaller features, better suited for segmentation. These can then be combined with k-means to provide a clustering. While in DSC [MKRLV22], they propose to combine DINO features with spectral clustering methods to provide a segmentation.

1.4.1 Neural Fields

A method which has taken over 3D Computer Vision is NeRF [MST+20]. The work itself aims to perform Novel View Synthesis given a set of images and camera poses. The system could be considered a form of SSL. However what is most interesting to us is their use of a Neural Field as the scene representation.

Neural Field is a coordinate conditioned neural network representing a field [XTS+21]. As previously indicated neural fields can be used as a parametrization for various signals, like RGB values [MST+20], signed distance functions [PFS+19], occupancy [SLOD21], albedo [SWL+21] or density [MST+20].

Usually the neural network used to represent these signals is a multi-layer perceptron (MLP). However the paper Deep Image Prior [UVL20], also uses a Neural Field for some of its in-painting tasks. However the function approximation is done with a CNN instead, processing a 2D grid of image coordinates at once. In this thesis we coin the term *Convolutional Neural Field* (CNF) to describe this type of an architecture.

The reason we are interested in Neural Fields is not in terms of their representational power, but rather their feature learning capabilities. By making a CNF reconstruct an image we are automatically defining a pretext task, which is useful for learning image features at multiple

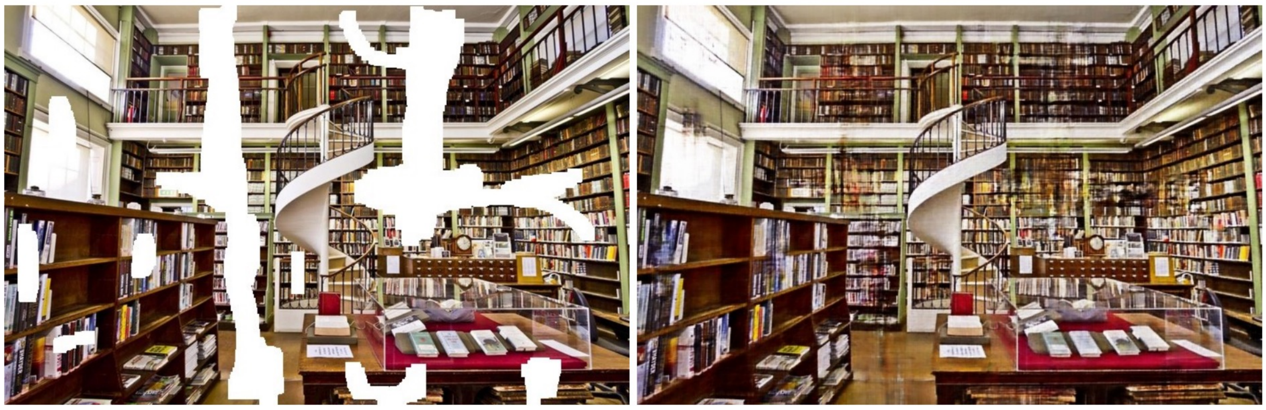


Figure 1.6: DIP [UVL20] inpainting results. Left: Original image with mask in white. Right: Results of inpainting with a CNF.

scales.

1.5 Motivation and Objectives

Due to Neural Fields being such a useful way to represent signals it is important to examine the features we learn from them.

In thesis we specifically concentrate on the task of segmentation. We attempt to explore how one could use a Neural Field trained to reconstruct an image to extract a segmentation in a self-supervised way.

However, we do not also want to rehash existing methods for clustering. It would be easy to take features from a CNF and use image clustering algorithms mentioned before. However these methods also do not scale to other domains, for example 3D. At the same time most vision systems are ultimately supposed to be used for real time inference, which is why we put an emphasis on efficient and online inference.

The contribution of this thesis is twofold, in the first section we concentrate on the well-defined [5] problem of interactive segmentation. More specifically we study how one can jointly

[5]This problem is well-defined, since we only have to consider the labels defined by the user at test time.

encode semantics and appearance in a CNF, using minimal user interactions to supervise the semantic information. This helps us study the features from CNFs and develop tools around these features. The problem of interactive segmentation is also important due to the aforementioned ambiguity of the problem definition.

In the section part of the thesis, motivated by our previous experience we first attempt to solve the dual problem of self-supervised contour detection using the features of a CNF. As previously mentioned the problem of contour detection has always gone hand in hand with segmentation, however, to the best of our knowledge we are the first work to propose self-supervised deep edge detection on images. We then use these self-supervised contours to extract a segmentation, with a novel method, reusing the aforementioned features.

Chapter 2

SegDIP

We now introduce SegDIP. In this section we develop a method for interactive segmentation using CNF features.

2.1 Introduction

As previously mentioned, the simplest problem definition for semantic segmentation is pixel-wise labelling. However this problem is fundamentally ill-defined, since these labels are non-unique e.g. we could label pixels of a brick in a building as both "brick" and "building". None of these answers are inherently wrong and the preferred answer undoubtedly depends on the application.

This is why typically for semantic segmentation we train deep neural network on large datasets, which pre-define the different types of classes. However, not only is there a high cost to creating these training datasets, but at test time we are limited to the classes we have previously seen. Instead, we explore an approach that allows for high-quality segmentation with minimal human interaction because the user monitors the semantic map as it updates in real-time and clicks only as needed to correct it. The smoothness properties of the CNN mean that regions and objects are coherently represented, and can frequently be labelled with only a few clicks. Sometimes, not even a single click is required as the correct properties will be transferred from already labelled parts of the scene.

Our contributions are as follows:

- We propose an architecture for real-time interactive semantic segmentation without prior data, a setup which has not been explored before.

- We demonstrate the powerful ability of the DIP architecture to predict semantics in an image by interpolating from just a few human provided clicks, without any explicit prior to do so. This is analyzed under the lens of the Neural Tangent Kernel.

- We propose a Temporal Consistency Loss, which takes a big step in helping our setup become a tool for large scale segmentation.

- We present applications of our system to video tracking, conditional image generation and unsupervised segmentation.

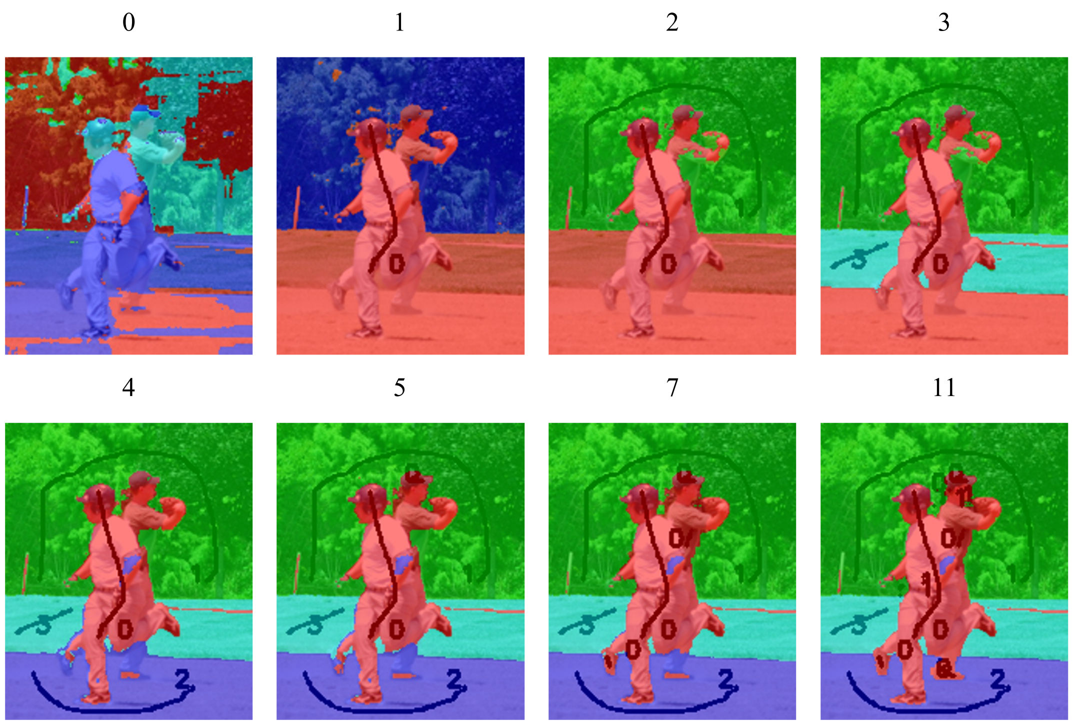


Figure 2.1: How sequential clicking works with SegDIP. We show the number of clicks supplied by the user and what the segmentation provided by the network looks like after them.

2.2 Related Work

2.2.1 Semantic Segmentation

Supervised Semantic Segmentation. Most standard semantic segmentation systems are based on training on densely labelled datasets without any further interaction during test time, for example U-Net [RFB15] or DeepLabV3 [CPSA17]. Such setups firstly require heavy resources for the training phase, they are inflexible to the addition of new unseen classes or refinement during test time and they require dense datasets which are often unavailable.

Semi-supervised learning. There have been methods aiming to alleviate these issues. ScribbleSup [LDJ+16] proposes training a network on a dataset labelled only with scribbles.

However this method again suffers from the issue of being inflexible to the addition of unseen classes at test time, due to the reliance on a training dataset. This is an issue which is seemingly overcome by both Polygon-RNN [Cas16] and Polygon-RNN++ [ALKF18], which are based on recurrent architectures with a human-in-the-loop who can adjust the segmentation at each step. However these methods again require expensive pretraining on a dataset. We argue that one should be able to do real time semantic segmentation without extensive pretraining just by analyzing the self similarity in a single image.

2.2.2 Single Image Based Learning

Recently, several deep internal learning methods have been proposed and achieve remarkable performance that is comparable to that of external methods trained on large-scale datasets. Methods have been developed to solve superresolution [SCI17] [UVL20] [BKSI20], restoration [SCI17] [MR19] reflection removal [YFC+20] or deblurring [RZW+20].

All these methods are predated by classical internal learning methods, which have proved internal approaches can be successfully applied to several image manipulation tasks such as super-resolution [GBI09] [HS12], dehazing [BI16] or texture synthesis [EL99].

2.2.3 Single Image Based Segmentation

There has also been some work proposed for segmentation. Double-DIP [GSI18] solves the issue of foreground/background segmentation using two Deep Image Prior architectures and a saliency initialization, we qualitatively compare against this work. However it is important to mention that despite requiring some interaction our setup allows for flexible multi class segmentation as opposed to being limited to two fixed classes. Furthermore we require no saliency initialization either.

There have also been classical internal methods for segmentation. Bagon et al. [BBI08] proposed an information theory method to define a good image segment and based on that segment an image. GrabCut [RKB04] proposed a foreground/background segmentation method based on graph cuts by using a user provided bounding box, specifying the region the foreground is in. We again use this work as a qualitative comparison baseline, however we again stress our setup allows for flexible multi class segmentation as opposed to being limited to two fixed classes.

2.3 Method

2.3.1 Deep Image Prior

Let's first revisit Deep Image Prior. As a general framework the paper treats a neural network f_θ with parameters θ as a parametrization $x = f_\theta(z)$ by mapping a fixed code $z \in \mathbb{R}^{H \times W \times L}$ to the image $x \in \mathbb{R}^{H \times W \times 3}$, where H and W are the height and width of the image.

With this general idea they are able to solve tasks such as denoising by taking the output of their network after utilizing early stopping to avoid overfitting and minimizing the objective:

$$\min_\theta ||x^* - x||^2$$

where $x^* = f_\theta$. A similar setup is used for inpainting, by applying a binary mask $M \in \{0, 1\}^{H \times W}$ representing the filled region and solving the objective:

$$\min_\theta ||x^* \times M - x \times M||^2$$

where $x^* = f_\theta$. The output of the network often predicts coherent textures on masked out areas. The function approximation f in both these cases is taken to be a convolutional neural network with the parameters θ representing the weights and bias in the filers of the network. The code z varies across tasks, but they are either meshgrid i.e. a grid of xy coordinates or random noise. For meshgrid we have $L = 2$, whereas for noise L can be arbitrary.

2.3.2 Basic Setup

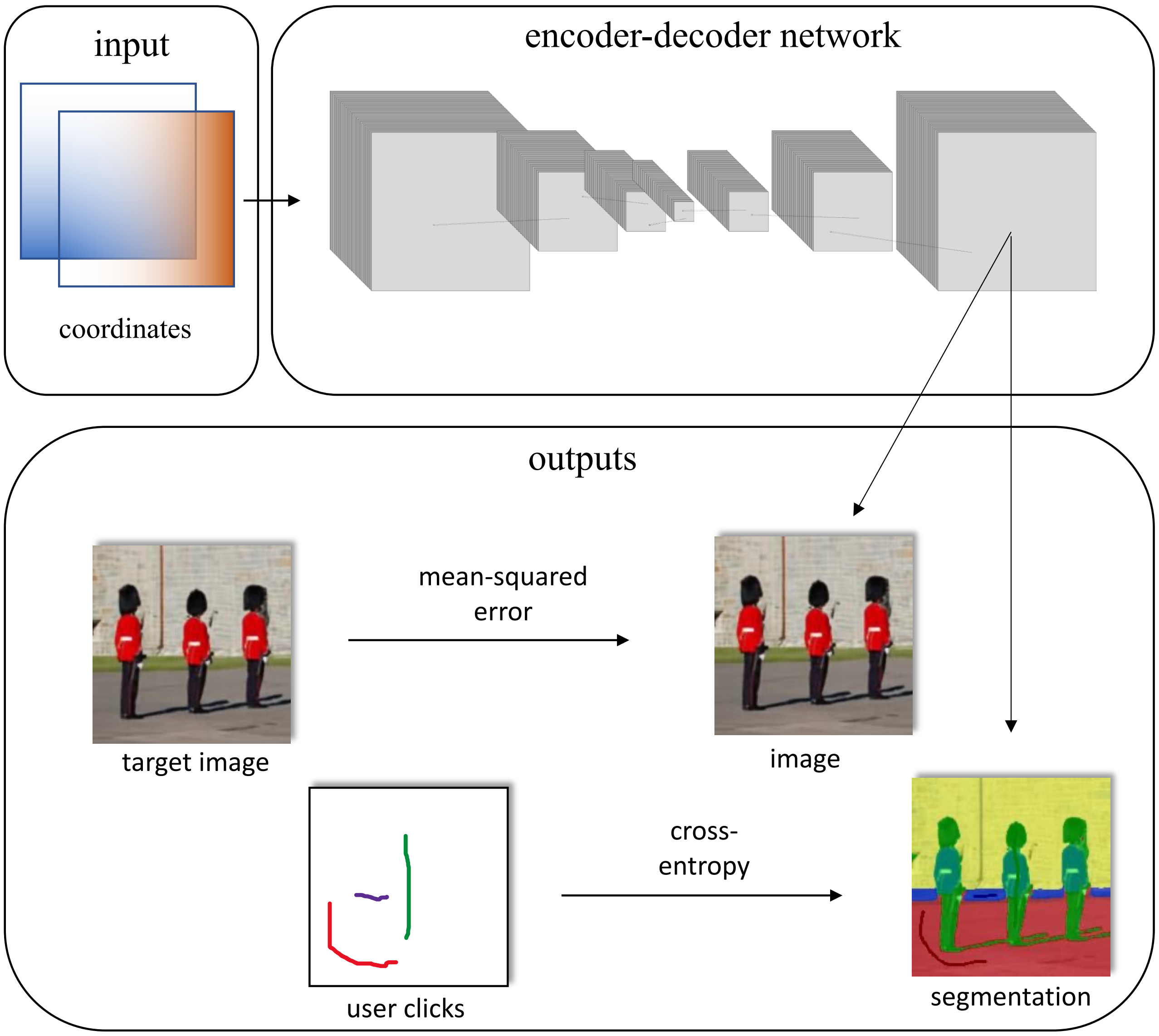


Figure 2.2: SegDIP architecture.

We augment this setup by first assuming we have sparse semantic supervision for the image. More specifically we have both a mask $M \in \{0,1\}^{H \times W}$, which masks out the pixels that do not have a ground truth annotation and a target segmentation $s \in \{0,1\}^{H \times W \times N}$, which contains the ground truth labels for the appropriate pixels contained in the mask M, where N represents the maximum number of classes we want to segment the image into. We then make the network map from the code $z \in \mathbb{R}^{H \times W \times L}$ to both the image $x \in \mathbb{R}^{H \times W \times 3}$ and the segmentation $s \in \mathbb{R}^{H \times W \times N}$ only for the annotated pixels according to the mask M i.e. we minimize the objective:

$$\min_{\theta} ||x^* - x||^2 + ||s^* \times M - s \times M||^2$$

where $x^*, s^* = f_\theta(z)$. This setup can be used for interpolating a few pre-defined annotations or for label denoising. However we further augment the system by real-time training and supplying clicks in the loop. In this case our objective at training timestep t is:

$$\min_{\theta} ||x^* - x||^2 + ||s^* \times M_t - s_t \times M_t||^2$$

where s_t represents the target with the ground-truth annotations supplied till timestep t and M_t is the binary mask containing only those pixels.

With this setup we can perform real-time semantic segmentation. We, however, find that this setup can sometimes degrade the segmentation of different areas with the addition of more annotations. This is why we introduce a novel Temporal Consistency Loss.

2.3.3 Temporal Consistency Loss

Let $Q = \{q_m\}_m$ be the set of pixels in the image given labels $\{y_m\}_m$ (probability vectors) respectively through an interaction (click, bounding box or stroke). Let S^k be the segmentation output of the network on the k-th training iteration of dimensions $H \times W \times N$, where H is the height of the image, W is the width of the image and N is the number of classes (each S^k_{ijl} is a probability). We define an augmented $H \times W \times N$ dimension matrix T^k as:

$$T_{ij}^k = \begin{cases} y_m, & \text{if } (i,j) = q_m \\ S_{ij}^k, & \text{if } (i,j) \notin Q \end{cases}$$

This is the current model predictions combined with the ground truth provided by the interactions. Furthermore let M^{k+1} be a matrix of dimensions $H \times W$, which has values between 0 and 1 of per pixel weighting.

Then at iteration (k+1), we define:

$$L_{k+1} = \sum_{i,j} \text{BCE}(S_{ij}^{k+1}, T_{ij}^k) * M_{ij}^{k+1}$$

where BCE is the binary cross entropy loss. Where we set:

$$M^{k+1}{}_{ij} = \begin{cases} 1, & \text{if } (i,j) \in Q \\ \lambda, & \text{if } (i,j) \notin Q \end{cases}$$

where $\lambda << 1$. We then at iteration $(k+1)$ jointly minimize:

$$\min_\theta ||x^* - x||^2 + L_{k+1}$$

only if there are some annotations in the image, otherwise, we only minimize the reconstruction loss, given by:

$$\min_\theta ||x^* - x||^2$$

With this setup we can segment many difficult images.

2.3.4 Implementation

We train the image reconstruction network for each image from scratch. In all our experiments we use an encoder-decoder network. The encoder has 4 layers, then we have a 3 layer decoder with 2 convolutional heads on top of that, one of which maps to the image and the other to the segmentation. Each convolutional layer has 128 filters and LeakyRELU activations. Between each of these layers we perform Batch Normalization [IS15], which we find to speed up the training process. For the input code z we use meshgrid i.e. xy coordinates, we find that this applies a smooth prior on the segmentation. Replacing meshgrid with random noise increases the capacity of the network to the point that it overfits particular annotations. To allow for a smooth training process and avoid jittering with the addition of new clicks we clip the gradients. The whole setup is trained with AMSGrad [RKK19]. We downsample through strides, but upsample with nearest-neighbour method.

For the setup with temporal consistency we use $\lambda = 0.001$. We train the network real-time with a Nvidia GeForce RTX 3080 GPU, which takes around 1 min depending on the image. We usually downscale images to have both dimensions in the size to be less than 300 pixels.

2.3.5 Training and annotating procedure

Before starting to annotate we usually wait for the image reconstruction to converge to the point that various objects in the image are recognizable. to start annotating a user has to first specify the class the annotation will be from, then he can use one of 3 types of annotations:

- Clicks - with which one can supervise one pixel
- Stroke - with which a user can more easily supervise pixels on one single line
- Bounding Box - a user can draw a bounding dox, with which he supervises all the pixels inside it

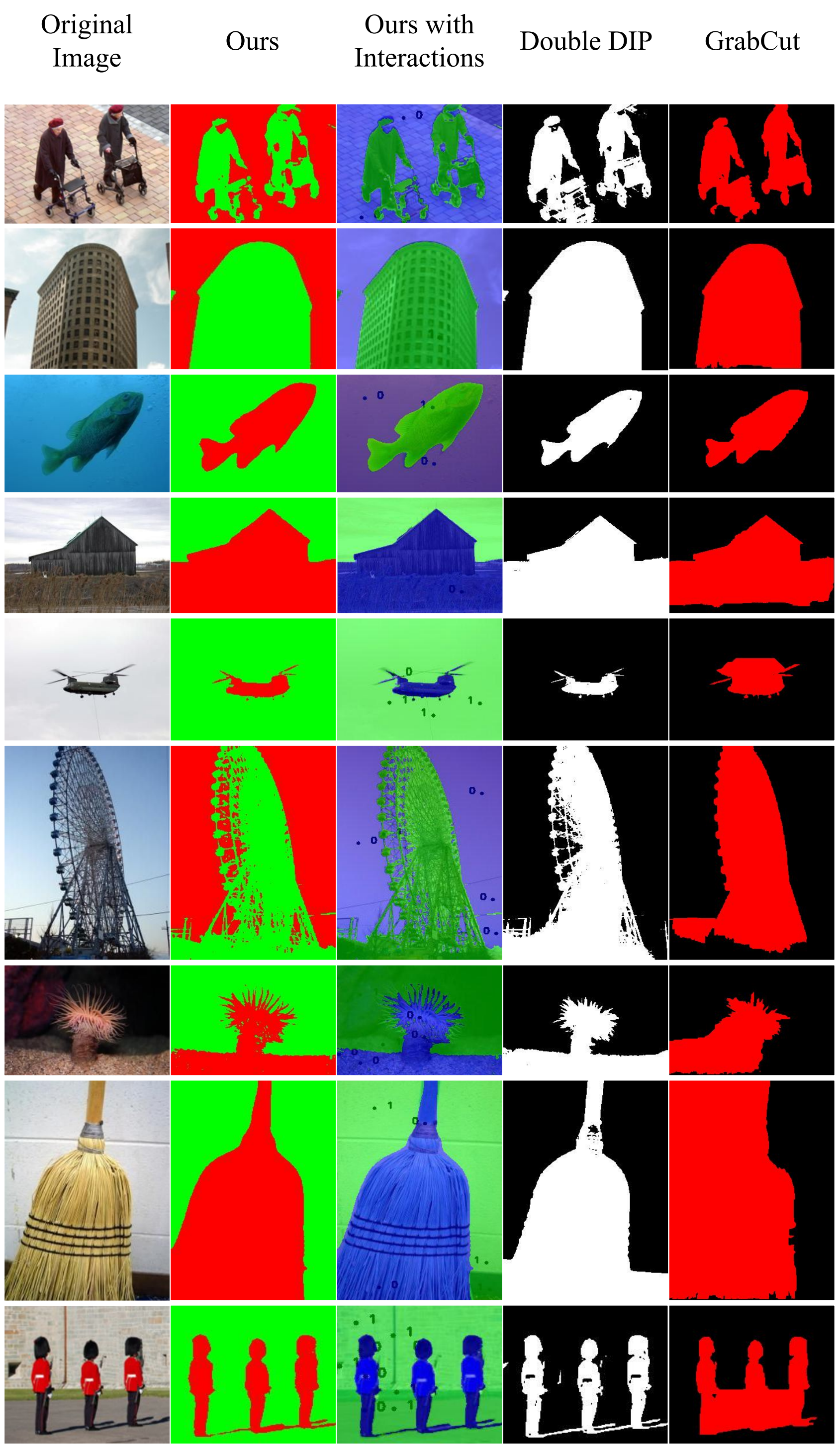


Figure 2.3: Comparison against Double-DIP [GSI18] and Grabcut [RKB04]. System without temporal consistency and with only click annotations. The number of clicks needed for us for the images is a) 2, b) 1, c) 3, d) 1, e) 4, f) 5, g) 6, h) 4, i) 7.

2.4 Results

We can firstly just observe how the sequential interactive segmentation looks like in Section 2.1. We can see that as the user provides the clicks the network itself find areas self-similar to the regions clicked and propagates the segmentation to those areas. The user can then simply fix any errors by the network by supplying more clicks in those areas.

2.4.1 Binary and Multi-Class Segmentation Without Temporal Consistency

We can first perform experiments without temporal consistency loss for 2 class segmentation on simple images with just clicking annotations (each click contains a single pixel). Even though this is not what we think is the key advantage of our system, it shows how well our architecture can interpolate from extremely sparse annotations. In Figure 2.3 we can see examples of this, where for each image we use at most 8 clicks, but still manage to produce a reasonable segmentation.

We can further see that on the qualitative examples our system outperforms GrabCut. We presume this is because of the very high sensitivity of GrabCut to the bounding box, for images with large foreground areas (with objects of interests) we see the performance is significantly worse.

Moreover we can see comparable results to Double-DIP despite not using any explicit priors for segmentation such as saliency. However we also "exceed" their performance on images with unclear foreground/background segmentation like the broom or on high frequency images such as the ferris wheel. This is due to the flexibility of providing supervision in our system - due to the interactions we are able to supply a user with their desired segmentation of the scene. Secondly we are abel to get high frequency details, because the segmentation is being performed from the same representation as the reconstruction, thus our system is able to see smaller details in images and recognize tiny patches.

What especially sets our system apart is the capability to perform multi class segmentation. This is also something both Double-DIP and Grabcut are not able to do. Results for this setup can be seen in Figure 2.4, where we perform multi-class segmentation from just a few user-provided clicks. We can again notice that for each of these image we need less than 5 clicks per class to produce reasonable segmentation.

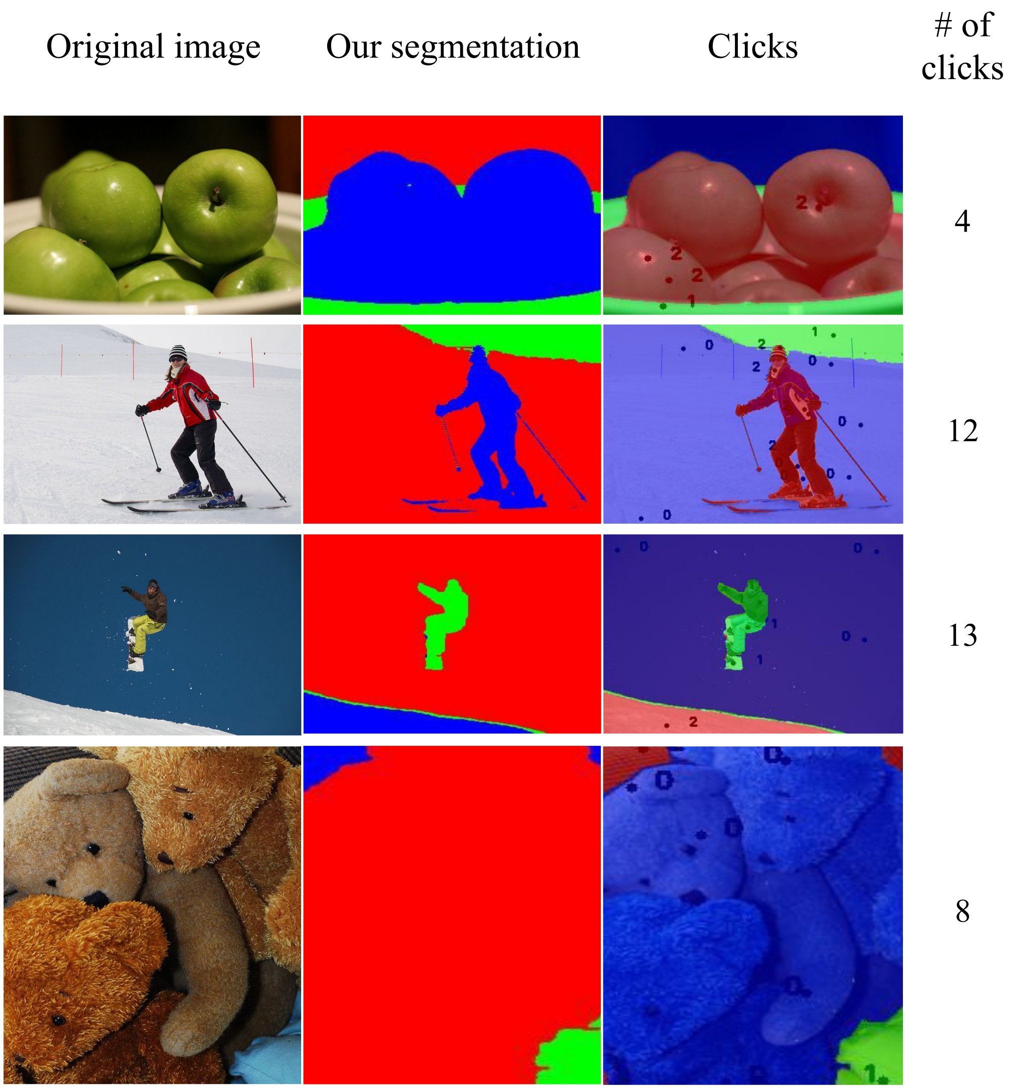


Figure 2.4: Segmentation from SegDIP on simple multi class images. All segmentation done in real-time with only clicking annotations, where click size is one pixel.

However the images we have shown so far are limited to simple or smooth textures. We have found for images with more complicated textures the segmentation quality sometimes degrades over-time, when a previously correctly predicted pixel is attributed the wrong class after the addition of more annotations. This in turns means we need a lot more annotations to

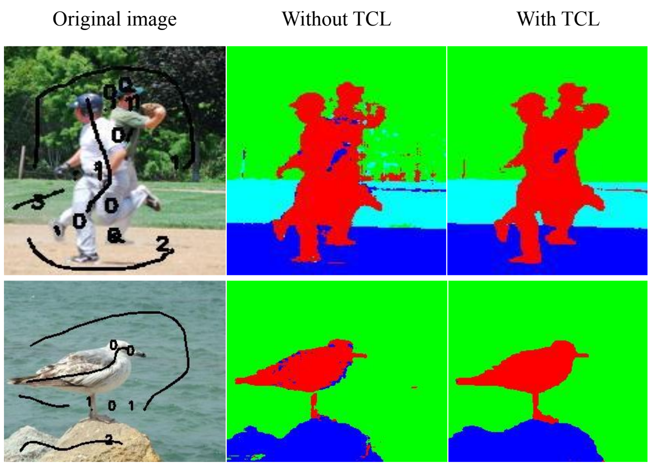
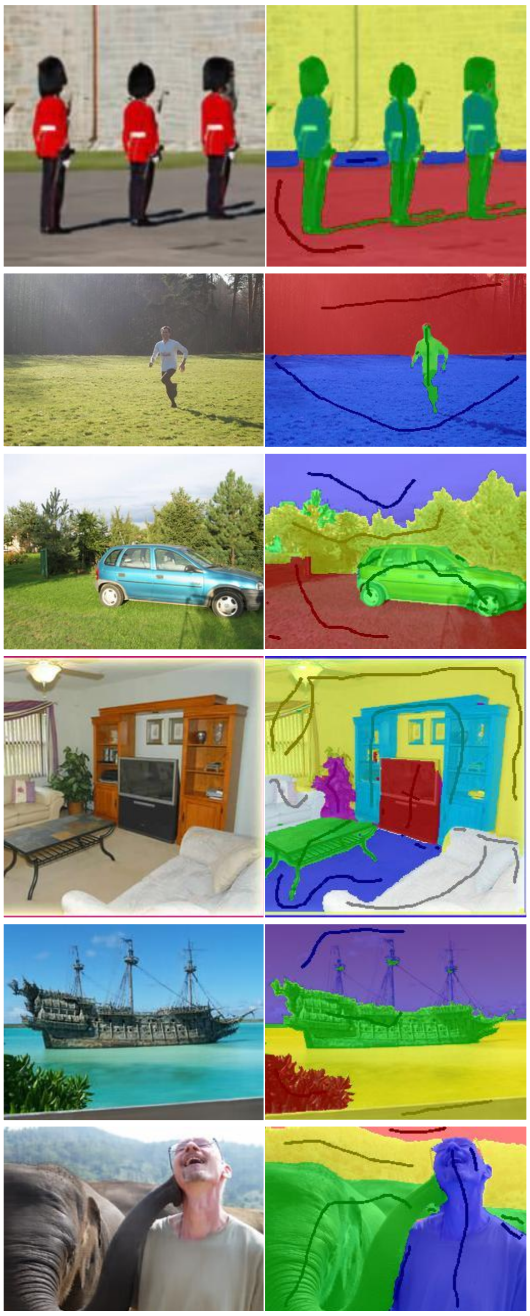


Figure 2.5: SegDIP for 3 class segmentation with and without the Temporal Consistency Loss (TCL). The number of user annotations for these images is a) 13, b) 6. Both these images were not segmented interactively, but run with pre-recorded clicks.

keep correcting previously correctly labelled pixels. This effect can also be seen in Figure 2.5. Where despite there being multiple annotations on the treeline the network gets confused by the highly textured leafs. As we can see on Figure 2.5 the Temporal Consistency Loss mitigates this issue.

2.4.2 Multi-Class Segmentation With Temporal Consistency

We can see further results on Figure 2.6, where we can segment complicated images with as little as 18 human provided strokes. However even more important is that for all but one of these images we are supervising less than 1% of the pixels in the image. Especially impressive are the results for highly textured images as the image with the car or the image with the elephant. Thus we can reach the conclusion that our system is very close to being a very efficient labelling tool. With this performance one could imagine the application of our setup for dataset creation. However it is also important to mention that the TCL suppresses label propagation slightly, hence why for simple images we might need more annotations than without it, however this issue is negligible in many real-world segmentation tasks.



Annotations	
scribbles	pixel (%)
3	0.35
8	0.41
13	0.77
18	1.12
4	0.37
10	0.81

Figure 2.6: SegDIP with Temporal Consistency Loss for multi-class segmentation trained in real-time with a user. Provided are the number of scribbles needed to segment the image in the particular examples and the percentage of pixels annotated by these scribbles. All but one of these images require less than 1% of pixel annotations.

2.5 Extensions

2.5.1 Neural Tangent Kernel

One can ask, why does this even work? We can further analyze the reason for the performance of the model, by studying the function approximation itself. The Neural Tangent Kernel [JGH18], shows that in the limit, where networks get infinitely wide, their behaviour can be approximated by a Gaussian Process. The kernel of this process is given by the Neural Tangent Kernel. Through this many important and interesting properties emerge. For example we are able to estimate the trajectory of the loss curve. However most importantly for us, we are able to define a similarity between training examples, for us this means between particular pixels of the image. This is done precisely through the tangent kernel, which defines a similarity between training samples (Gram matrix).

Neural Tangent Kernel methods have been used to analyze the positional encoding for neural representations [TSM+20], optimizing networks for shape completion [CPW21] and similarly for Neural Architecture Search [CGW21], reducing required GPU hours. Most importantly for us it has been used to analyze the performance of Deep Image Prior for denoising [TTD21]. Interestingly the paper finds that the DIP with normal Stochastic Gradient Descent optimizier is like a non-local filter and the behaviour of the network can be approximated by some fixed kernel. While with ADAM [KB14] it acts like a non-local filter, improving performance.

In this section, we thus consider our network trained only with a reconstruction loss, without any interaction. We can thus analyze what kind of a prior (kernel) is imposed by our network and how this effects the segmentation.

For Gradient Descent and with the infinite depth assumption for each layer the kernel is constant. However this changes when we have a rather small depth at each layer and when we use Adam optimizer [KB14]. At training iteration t, Adam optimizer updates the weights according to:

$$\theta^{t+1} = \theta^{t} - \tilde{\eta} H^{t} \frac{\delta \mathcal{L}}{\delta \theta} \left(\theta^{t}\right) + \beta_1 \left(\theta^{t} - \theta^{t-1}\right)$$

where θ are the weights of the network β_1 is a hyperparameter controlling the momentum and learning rate, $\tilde{\eta} = \eta (1 - \beta_1)$ and H^t is a diagonal matrix containing the inverse of a running average of the squared value of the gradients, computed using the hyperparameter β_2. The network outputs are then updated according to:

$$z^{t+1} = z^{t} + \tilde{\eta} \tilde{\Theta}_{L}^{t} \left(y - z^{t}\right) + \beta_1 \left(z^{t} - z^{t-1}\right)$$

where for simplicity we used $z = x^*$ and where $\tilde{\Theta}^t$ is the resulting gram matrix, which is adapted at each iteration, according to the rule [TTD21]:

$$\tilde{\Theta}^{t} = \left. \frac{\delta x^*}{\delta \theta} H^{t} \left(\frac{\delta x^*}{\delta \theta}\right)^{\top} \right|_{\theta = \theta^{t}}$$

where $x^* = f_\theta(z)$. The matrix H^k imposes a metric in the weight space which differs from the standard Euclidean metric of Gradient Descent.

The calculation for the whole gram matrix is very expensive and scales quaternarily with the number of pixels. Thus we will instead pick a single pixel and find the affinities with other pixels as per the aforementioned rule. It is important to note that the kernel is not stationary and changes over the training process.

In Figure 2.7 we can see how the gram matrix for a given pixel changes as the training progresses. We can see that at the start of the training high affinity areas are vaguely clustered around the pixel in question. However these high affinity areas become more semantically meaningful as the training continues. We can also see how the affinity map becomes more concentrated around specific pixels with more iterations, this is visible in the bird example and one of the living room examples. This in turn also means a higher intensity

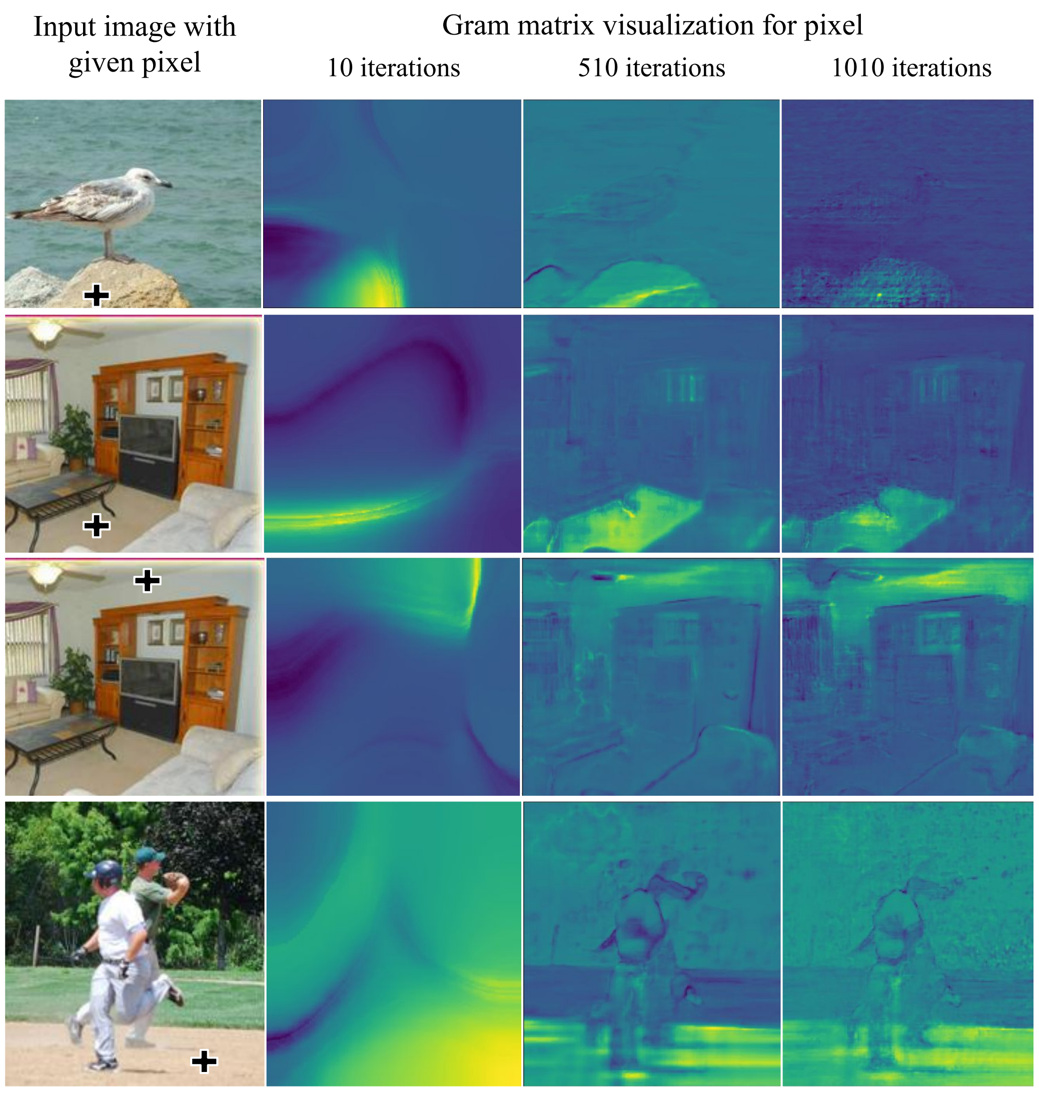


Figure 2.7: Gram matrix visualized for specific pixels in an image based on the Neural Tangent Kernel at different iterations of the network. The reference pixel given in the first columns. Brighter regions have higher affinity.

on some small specific areas.

The kernel map intuitively predicts how the labels will propagate for a single click in that area, by denoting the similarity between the pixels. Hence it also makes sense that we would want to start clicking before the network overfits to the image (iteration 1010).

It is interesting to notice that the kernel seems to be doing more than just simply colour based intensity mapping. This is especially visible on the living room picture, where we visualize the similarities for a pixel on the wall. There are parts of wall separated by the furniture that is not attributed a high affinity despite having a very similar colour to the highlighted pixel. We can thus conclude that our network imposes a similarity based not only on colours, but also the geometry of the image itself.

2.5.2 Video Tracking

Video tracking is the problem of segmenting the same object across different frames in a video of a scene. Past methods used to tackle this problem include using rigid registration [BR08], mean shift [CRM00] or deep learning based methods [LMS+22][1].

We mend SegDIP to having an object tracking capability. Intuitively we segment the first frame of the video, while optimizing our network to reconstruct it. Later on we change the frames to optimize the reconstruction for, to subsequent frames, while the video segmentation consistency remains. The setup is similar to the concurrent work of Lei et al. [LXOC22], however we use interactions as opposed to a dense groundtruth map for the scene. On top of that our method is able to work online.

Let $\{x_t\}_t$ be a set of frames from a video, where $x_t \in \mathbb{R}^{H\times W\times 3}$. For tracking in a video on the first frame we train the exact same setup as with the Temporal Consistency Loss, allowing a user to annotate the frame in real time, i.e. if there are annotations in the frame we are minimizing:

[1]This is not a comprehensive review of video tracking.

$$\min_{\theta} ||x^{*} - x_1||^2 + L_{k+1}$$

However now we allow the user to switch the frame of the video to the next one, retraining the reconstruction loss towards the new image target. In this way if the user switched the loss towards the frame t before timestep $(k+1)$ then we are minimizing:

$$\min_{\theta} ||x^{*} - x_0||^2 + L_{k+1}$$

however it is important to note that we reinstantiate the aforementioned annotations Q for each frame, hence we have:

$$L_{k+1} = \sum_{i,j} \mathrm{BCE}(S_{ij}^{k+1}, T_{ij}^{k}) * M_{ij}^{k+1}$$

where ${M^{k+1}}_{ij} = \lambda$. This forces also a temporal consistency across different frames on the video. We note that this setup let's us track the objects we annotated only in the first frame across the whole video.

We can perform video tracking experiments on two simple scenes. We switch the frames every some iterations (as described above). For each video we only provide annotations in the first frame. From Figure 2.8 we can see that SegDIP performs well on these simple tracking problems, it does not get pixel perfect results, however it is able to generally track the shape as it moves through the scene.

Our setup also allows fixing errors during the training in further frames, however we concentrate on only showing tracking abilities in these experiments.

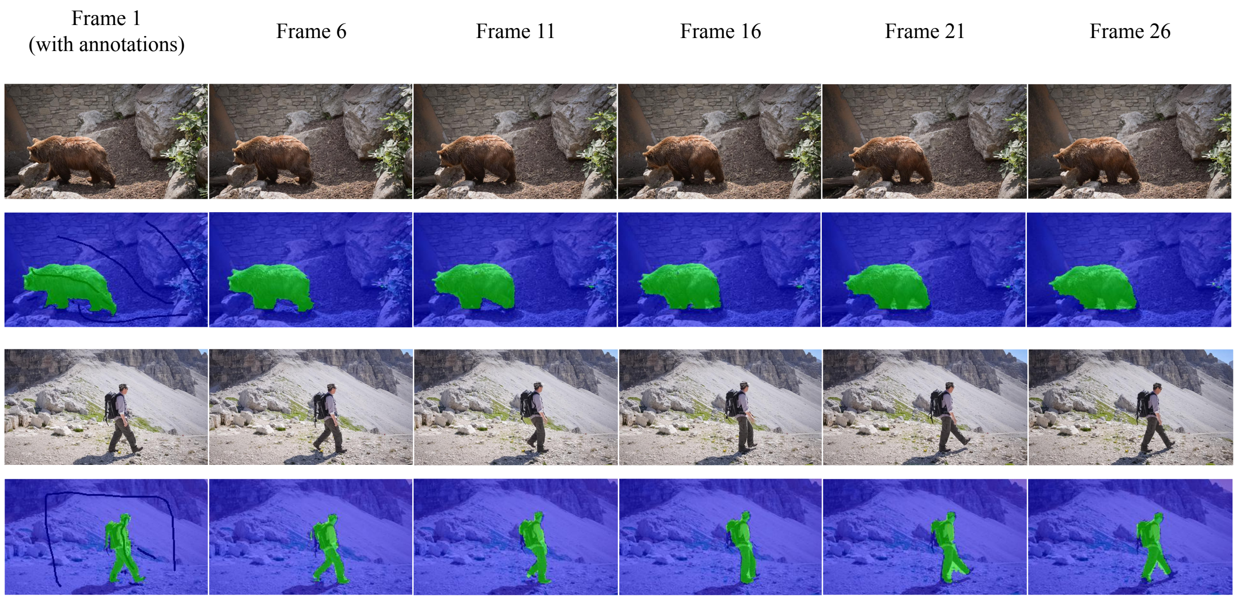


Figure 2.8: Tracking results for with SegDIP. We can see that annotations given on the first frame and no other annotations on the subsequent ones. Even with videos longer than 20 frames we can see the the inter-frame consistency remains.

2.5.3 Image Generation

Image generation from a single image has been attempted before. The most famous work in this regard is of course SinGAN [RSDM19]. The paper trains a multi-scale GAN, to learn to generate images, with training the system on a single image. Similar work recently replaces the GAN with a more classical patch based method, which is more stable [GFS+22].

In SegDIP, we are using an image and sparse annotations to infer a dense segmentation. Inspired by the cosegmentation setup in [BBI08], we could ask whether we can use the segmentation to generate an image. In this spirit, suppose we are given an image. We then use SegDIP to segment this image with sparse annotations, but at the same time we also draw semantic annotations on a "blank canvas", which is then used to generate a new image.

More formally suppose we have our target image $x \in \mathbb{R}^{H \times W \times 3}$, which we augment by making it two times wider i.e. $x \in \mathbb{R}^{H \times 2W \times 3}$, where we just add an array of zeros on the right side. We then make the network map from the code $z \in \mathbb{R}^{H \times 2W \times L}$ to both the image $x \in \mathbb{R}^{H \times 2W \times 3}$

and the segmentation $s \in \mathbb{R}^{H\times 2W\times N}$, however the loss for the image is only applied to the left side of the x tensor. In other words suppose $M^* \in \mathbb{R}^{H\times 2W}$, such that:

$$M^*_{mn} = \begin{cases} 1, \text{ if } n \leq H \\ 0, \text{ if } n > H \end{cases}$$

Then we also have the mask from our segmentation, as before, just doubled in width, it is indexed by time, due to the online user interaction:

$$\min_{\theta} ||M^* \otimes (x^* - x)||^2 + ||s^* \times M_t - s_t \times M_t||^2$$

where s_t represents the target with the ground-truth annotations supplied till timestep t and M_t is the binary mask containing only those pixels.

The results of segmenting a soldier image and generating a image from a similar distribution is presented in Figure 2.9. There are 6 classes present in the given example ground, grass, wall, pants, shirt and helmet.

A few things are especially astonishing. Firstly the generated image as emergent classes present in it. The gloves and the shadows are not classes present in the semantic image provided to the image network, nonetheless they are present in the generated image. Clearly this single image was enough for the network to learn the pattern of a glove present by the intersection of the pants and shirt or there is a shadow cast by the person.

Secondly we can see that the network is not just generating colours, but also textures. This is better visible on the grass, where the grass is two toned on the original image and it is also two toned in the generated image. This result shows that the network does not just have colour information, but textural information, which it also uses for segmentation tasks.

These results are really astonishing and show akin to SinGAN [RSDM19] how much can be learnt from a single image.

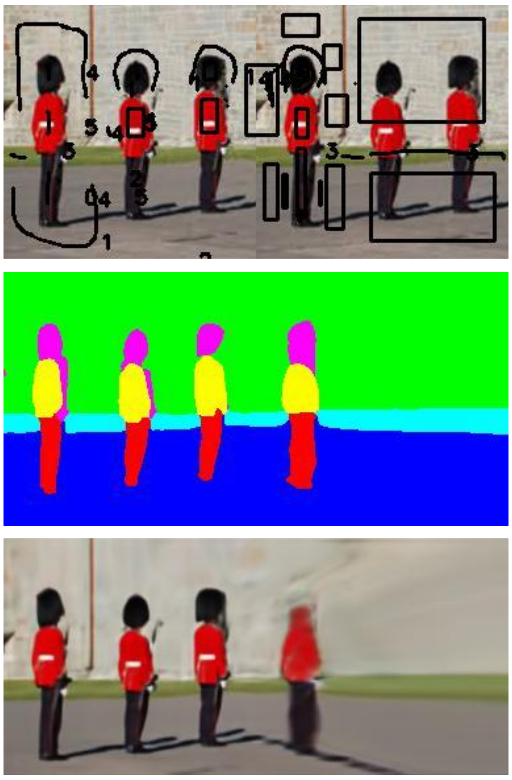


Figure 2.9: First row: the original image overlaid with the annotations (box ones fill the whole region), second row: shows the interpolated segmentation from the network, third row: shows the image output from the network.

There are of course drawbacks, for example the helmet is not present in the generated image, despite the annotation give. This is supposedly due to the helmet class leaking to the shirt (as seen on the segmentation).

Nonetheless this defines a very interesting experiment, which could be taken further for image generation using a single image/scene.

2.5.4 Self-Supervised Segmentation through self-distillation

The final extension for SegDIP is an idea for self-supervised segmentation i.e. segmentation without any user input. This is of course not well defined, however it would be very interesting to see what kind of a segmentation the network itself generates, given that we know from earlier that it has a textural understanding of the image.

The setup we will present is similar to region growing segmentation methods, which were highly prevalent in the literature before [BJ88].

Inspired by the Temporal Consistency loss, we supervise the network through its own predictions. More specifically, suppose our segmentation has n classes, each initialized with a separate click as in SegDIP. Then at each training iteration for each class we find the pixel with the highest probability to be in that class and provide a click on it automatically (with the label it has the highest probability for).

More formally let our loss be as before:

$$\min_{\theta} ||(x^* - x)||^2 + ||s^* \times M_t - s_t \times M_t||^2$$

This time however for each n classes we at t-th iteration we find the pixels $q_1, q_2 ... q_n$ with the highest probability (as per the networks own prediction) of being in the subsequent class we replace $s^*_{q_i,i} = 1$ and $M_{t_{q_i}} = 1$.

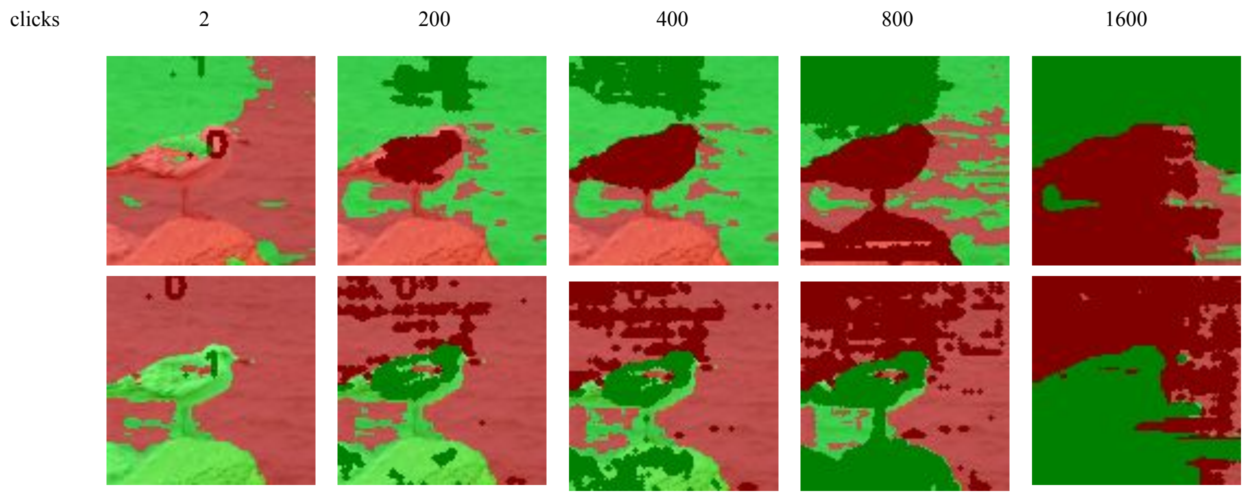


Figure 2.10: First row: region growing with spread only to neighbouring pixels, second row: label propagation with non-restricted spread

Results of this can be seen in Figure 2.10. The results are not similar to how a human would segment the images and is even worse than colour segmentation.

The following formulation has an issue that we are not adding anymore information to the segmentation with these clicks. This is mainly because of the overconfidence of the network. As can be seen, the network is highly overconfident of its own predictions, and hence a new click will not change the expected distribution of the network.

We thus also try a setup where the labels can only propagate to the nearest regions. This adds a sort of "coherence prior", where pixels nearly are most likely to be in the region being grown. This is clearly not scalable to disjoint objects and struggles with its own issues of when to stop region growing. A simple threshold version, where we stop region growing once there are not enough pixels with a high probability to belong to a certain class does not work due to the overconfidence of the network.

2.6 Conclusions

We have shown that online, image-specific training of a compact CNN model which jointly encodes appearance and semantics allows ultra-sparse interactive labelling to produce accurate dense semantic segmentation. Despite promising results, our system's label propagation mechanism works well mainly for proximal regions or those sharing similar colour/texture.

However it must be mentioned that the setup we are exploring is extremely constrained and apart from colour and proximity, there is not much more information that can be found through a single image. Nonetheless we expect a deeper understanding and study of network architecture and its biases would give us a lot more insight into engineering further segmentation priors into the network itself e.g. through architectural changes.

Nonetheless the system is very versatile and introduced a lot of new ideas, also some further explored in the subsequent section of this thesis.

Many of the extensions offer exciting new ideas and possibilities for further work. We are especially astonished by the video tracking results and even though they are still far from pixel perfect tracking, we believe the results have a lot of potential for improvement. Some further areas of exploration could include incorporating optical flow information into the network or helping the network converge quicker to the new frame.

The image generation results are also very exciting and show a glimpse of the understanding the network develops of the image.

Chapter 3

Self-Supervised Segmentation using Contours

We now move on to the problem of self-supervised segmentation from CNF features. From the previous section it was visible that the single segmentation head on top of the CNF features gives unreliable uncertainty estimates and we can not derive a meaningful segmentations from it.

In this work we thus concentrate on an ensambling type method, which tests multiple hypotheses for segmentations and combines all of them together through the idea of contours.

It is important to mention that for self-supervised segmentations it is not obvious how to combine multiple maps, since the labels do not mean anything. It is more sensible to combine *contour* maps as they are label agnostic. This motivates our work using contours.

3.1 Introduction

There has been a major uptrend in methods for self-supervised segmentation due to the increasing quality of self-supervised image features. Most of these methods consider pretext tasks on large datasets to learn useful feature extractors, which can then be used for various downstream applications.

Single scene learning, however, has seen an abundance of research with the advent of Neural Representations. Parametrizing scenes and images using neural networks is a natural way to learn hierarchical features describing the data.

In our work we show that Neural Representations, due to their reconstructive nature, encode features useful for self-supervised segmentation.

We design a way to use a Neural Representation to extract image contours, a problem dual to the task of segmentation. We then describe a method to go from contours to a segmentation, all within the scope of the same Neural Representation.

More specifically we make use of a *Convolutional Neural Field* (CNF) to map from an image of coordinates to the target RGB image and multiple segmentation heads, which all explore various segmentation hypotheses. We then design a custom loss, which distills the collective information from each head across all of them to reach consensus contours. On top of the network we have another segmentation head, which learns to match the contours found with the aforementioned method and produce a final segmentation.

Our system outperforms classical methods for self-supervised contour detection, which make use of hand designed features. We also show that qualitatively the segmentations from our method are of similar quality to those produced by state-of-the-art self-supervised segmentation methods. Our experiments showcase that there is a lot to be learnt from just relying on self-similarity within a single image.

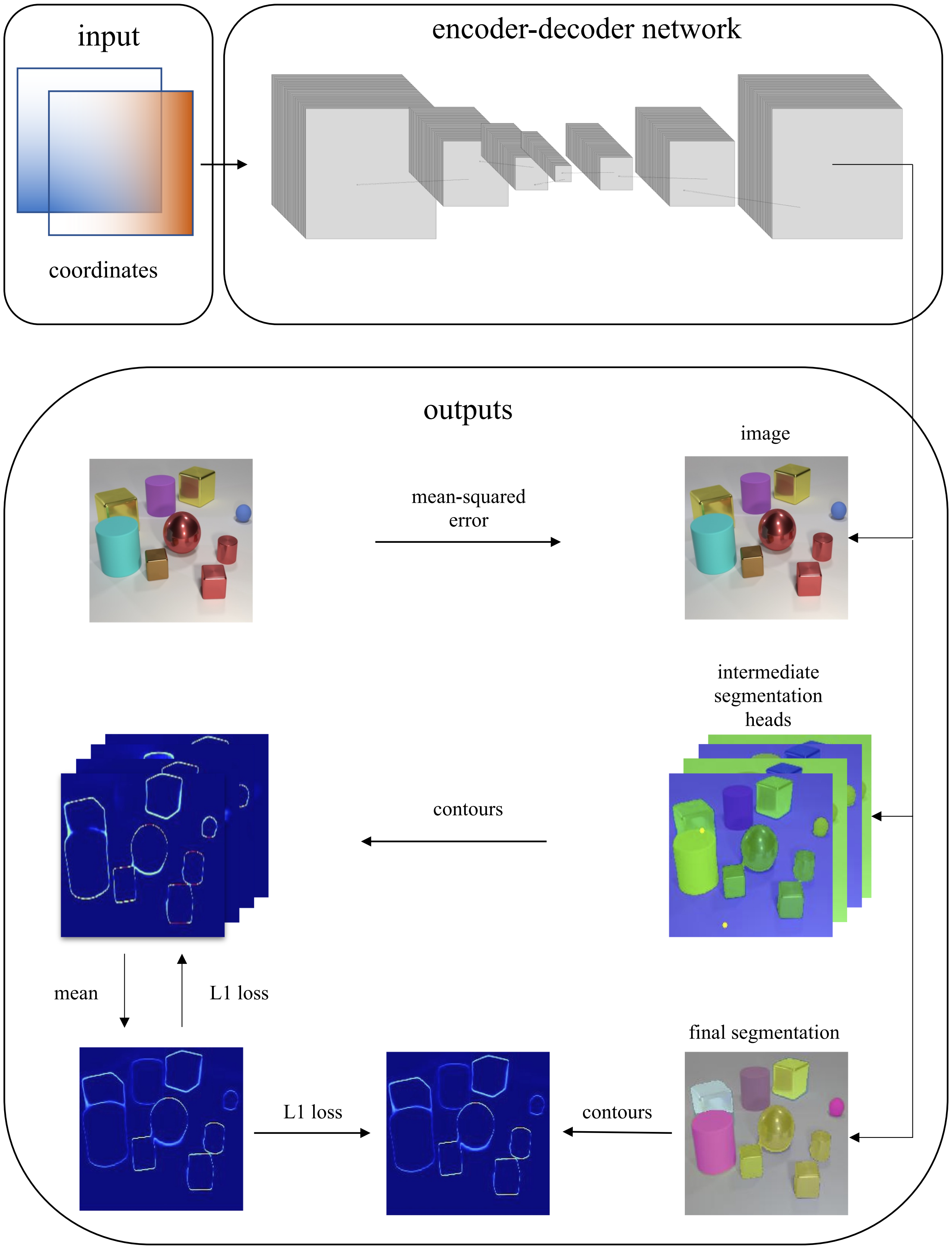


Figure 3.1: System diagram for our contour detection method. Our coordinate conditioned encoder-decoder network produces three outputs. The original image, trained with a mean-squared error for reconstruction. Multiple intermediate segmentation heads, trained with self-distillation loss. Additionally the final segmentation, trained to match the mean contours.

3.2 Related Work

3.2.1 Self-Supervised Feature Learning and Segmentation

Non-Learning methods. There has been a lot of prior work on single image segmentation without the use of deep learning. Normalized Cuts [SM00] forms the problem of segmentation as a graph-cut problem on a weighted graph where the nodes represent the pixels and the weights between them are affinities, found by composing exponentials with distance functions of coordinates or intensities. Mean Shift [CM02] uses a mode seeking algorithm to find the pixel modes in feature space. All these algorithms use hand-designed features to represent the image.

Deep Learning methods. Self-supervised feature learning [HFW+20], [GSA+20], [CTM+21], [CKNH20] has been essential for the advent of self-supervised segmentation methods. An example of such work is DINO [CTM+21], the work proposes to learn useful features by making a teacher and student network map augmented image patches to a distribution, essentially defining a dummy classification task on patches.

DINO has sparked many works on self-supervised segmentation [HZH+22], [WSH+22], [MKRLV22]. Most of these methods reuse couple classical clustering methods like spectral clustering, k-means clustering or normalized cuts with features learnt in self-supervised learning frameworks.

Due to the features being trained on datasets, they are able to perform object recognition as well as segmentation. In our method we are not able to generalize across images and hence lack the ability to perform recognition. However our total training time (including learning features) is greatly reduced by training on just one image.

Furthermore our method showcases how much one can learn from processing just a single image using deep priors.

There has been some previous work also on learning features from a single image and using this for self-supervised segmentation [KKT20] or superpixels segmentation [Suz20].

Due to supervision only trough reconstruction of the signal, however, our method can generalize to various domains, for example 3D, since our inputs are pixel coordinates instead of an actual image. The processing we do, could be instead done with an MLP.

3.2.2 Contour detection

In our work work we include the intermediate step of contour detection. There have been several previous methods proposed for contour detection. Earlier methods relied on convolving gradient filters with grayscale images like Roberts [Rob63] and Sobel [Sob14]. A similar approach is proposed by the popular Canny edge detector [Can86], which convolves a gradient filter with a grayscale image, but additionally uses non-maximal suppression to get thinner edges and hysteresis for continuity in edge strengths.

After the advent of labelled datasets [MFTM01], methods have relied on supervision to learn the contour detectors BEL [DTB06], Pb [MFM04], gPb [AMFM11], SED [DZ14]. As an example Pb [MFM04] uses local oriented gradient filters, which are then used as pixel descriptors for a logistic regression solver, gPb [AMFM11] adds an extra global feature to the weights using spectral decomposition.

With the uptake of Deep Learning, methods have still mainly relied on supervision for contour prediction, however the features used for these edge detectors are now learnt DeepEdge [BST14], DeepContour [SWW+15], HED [XT15], COB [MPTAG17].

As previously mentioned the features are learnt from a dataset with groundtruth contours given. To the best of our knowledge we are the only method for generating self-supervised deep contours.

3.3 Method

Our method requires a target image $x \in \mathbb{R}^{H \times W \times 3}$. We perform image specific training and require no pretraining.

3.3.1 Overview of Method

In short we have a Convolutional Neural Network f_θ which takes in an image of coordinates and outputs features $f \in \mathbb{R}^{H \times W \times 128}$. We map these features to the **original image** x with $f_{\theta'}$, N **intermediate segmentations** with f_H and a **final segmentation** with $f_{H'}$. The network f_θ with $f_{\theta'}$ is trained only with the reconstruction loss for the original image. The layer f_H is trained to match randomly sampled ground truth points for each segmentation and a contour based **mutual distillation loss** between the heads. The final segmentation head $f_{H'}$ is meant to match the contours found using the aforementioned loss. An overview of the method can be found in Figure 3.1

3.3.2 Intermediate Segmentations

The feature f from the network f_θ is transformed to N, m class segmentations i.e. N features of size $H \times W \times m$, where each position spatial position defines a probability vector over m classes. To ensure variance across each of these N heads, for each head we pick random m separate pixels, for which we randomly assign to one of the m classes. This is enforced by adding a cross entropy loss on the chosen pixels for each head.

This loss ensures each head is concentrating on separating different parts of the image. Furthermore after some iterations to help the heads find a consensus segmentation we combine them. Combining them simply through a mean would not work, since the classes in each head have been assigned randomly.

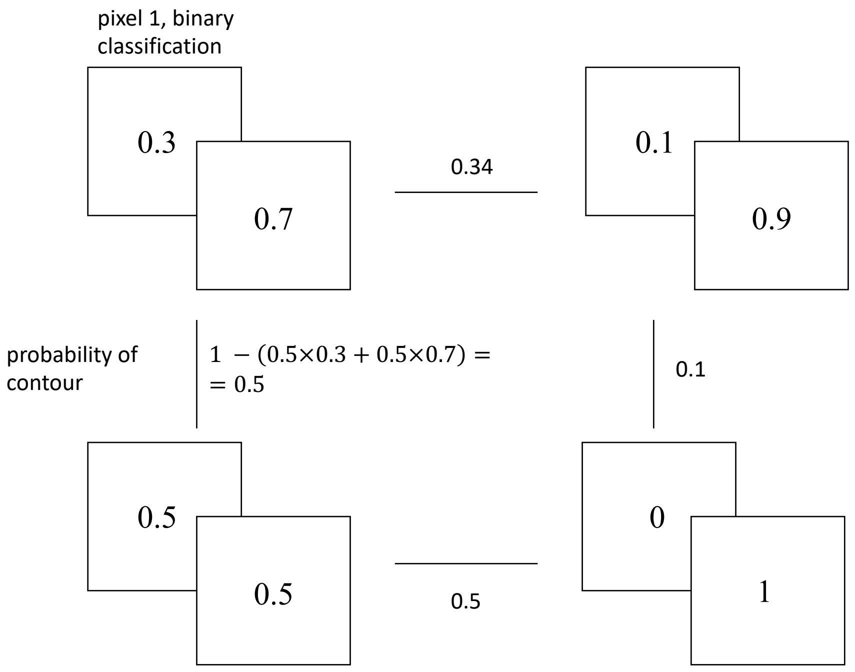


Figure 3.2: Depiction of the differentiable transformation from segmentation to contours.

Instead we combine the heads through a mean of contours. Given a segmentation head i.e. $H \times W$ m class distributions, we take neighbouring pixels and we find the probability they are in different classes, this gives the probability of there being a contour between them. This is illustrated for a 2×2 sized with a 2 class segmentation in Figure 3.2.

We can see this transformation from segmentation to contours is differentiable and it is class agnostic, hence we can take of these contours across all heads to get a mean contour image as depicted in Figure 3.1.

We then take the L_1 loss between the contour image from each segmentation head and the mean edge image and propagate this loss through to f_H. This ensures a consensus is reached between each of the heads, while the aforementioned clicks ensure our heads are diverse.

3.3.3 Final segmentation

For the final segmentation we again find its contours using the idea from Figure 3.2. We then again use an L_1 loss to have the final segmentation match the edges from the mean edges.

To ensure this final segmentation does not collapse to a few classes we use centring before applying the softmax, this is the same idea as used in DINO [CTM+21].

3.3.4 Implementation details

For f_θ we use a CNN encoder-decoder with downsampling using strided convolutions in the encoder and upsampling using nearest neighbour upsampler. For the means retention value in the centring we use 0.2.

We also use $m = 4$, due to the 4 colour theorem [AH77]. We found that generally many of the heads after the mutual distillation loss converged to the same tensor, which is why we needed them to be able to represent any possible edge configuration. The 4 colour theorem states that any planar graph can be coloured with 4 colours, hence by having 4 classes we can represent any possible edge image with a single head not restricting our convergence.

All the experiments were conducted on an Nvidia GeForce RTX 3080M with 16GB of memory.

3.3.5 Connection to Normalized Cuts

The method we have is an extremely unique setup for self-supervised segmentation and there is not many direct connections one could make. However there are some high-level connections we could make, to help understand the method.

Contours as affinity. Firstly the contour map we have can be treated as a probability map i.e. the probability that two pixels are a different class. This can also be treated as a distance measure between the two pixels. Hence as in Normalized Cuts we are constructing a weighted graph on the pixels space. However, we are only calculating the distance between each two pixels nearby. Hence in essence we are constructing a simplified, more memory efficient graph, which reduces to nearby edges. This can be interpreted as having edges only for neighbours with L_1 distance 1 in coordinate space. This is better illustrated in Figure 3.3.

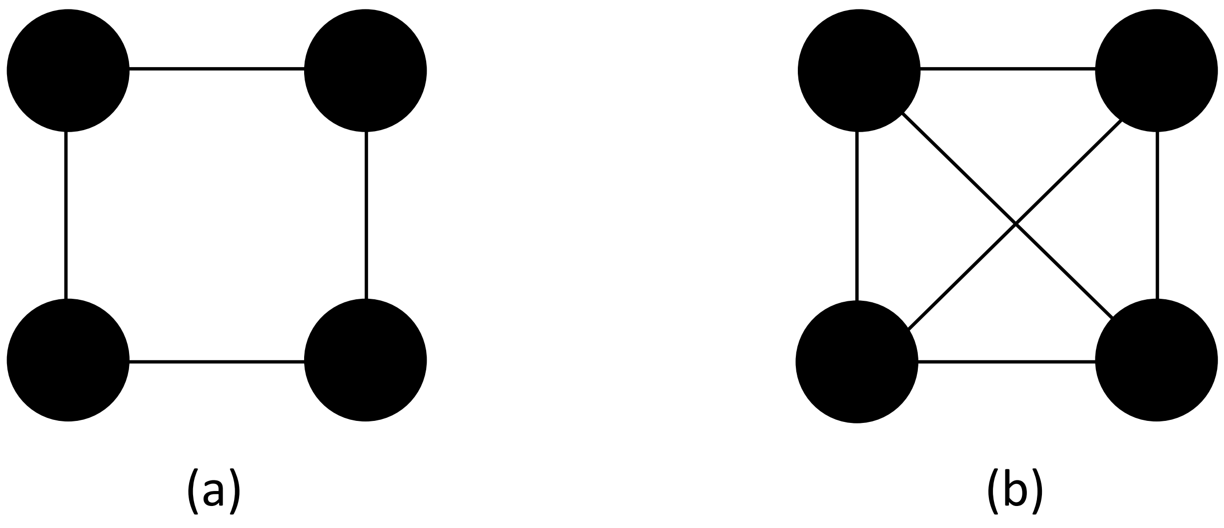


Figure 3.3: (a) The construction of our graph is reduced to edges between neighbours. (b) Normalized Cuts [SM00] assumes a fully connected graph.

Distance bias. Furthermore another connection can be made in their features. In Normalized Cuts, the authors explicitly add the coordinates as features describing pixels, we do something similar by having the pixels as input to the network, biasing the network to have similar predictions for nearby pixels.

3.4 Results

We first perform contour evaluation on the Berkeley Segmentation Dataset (BSDS500) [MFTM01]. More specifically we use the 200 training images given in the test set for benchmarking. We compare against other contour detection methods. For the contour experiments we will use non-maximal suppression [Can86] for thinner contours.

The method we use for comparison is the precision-recall framework developed by Martin et al. [MFM04]. The Berkeley dataset contains 5 groundtruth edge maps drawn/developed by separate users. The comparison takes a contour map, which gives each pixel a probability of

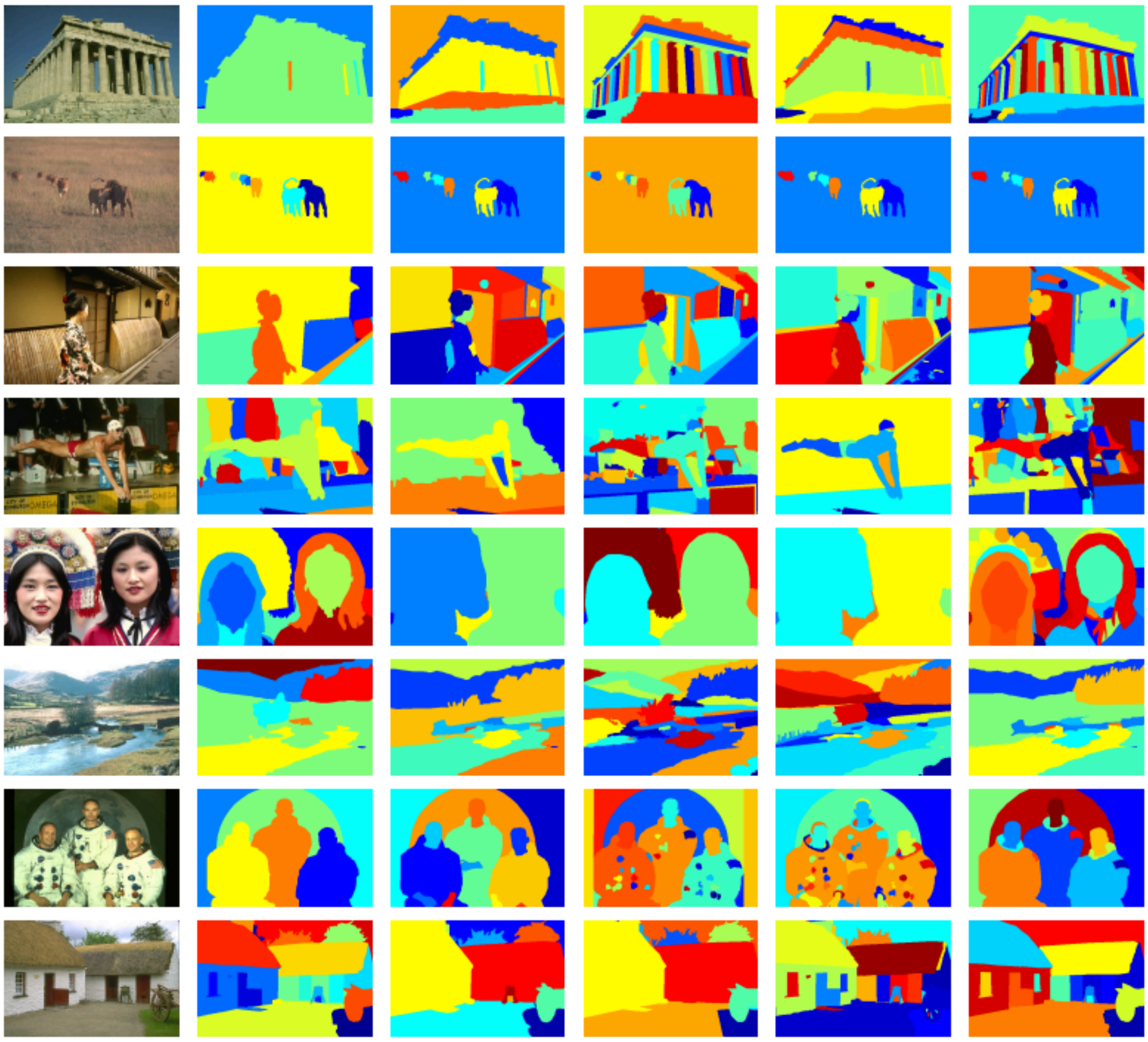


Figure 3.4: Groundtruth image alongside 5 human drawn segmentations from Berkeley Segmentation Dataset [MFTM01]. Figure redacted from [Pan08].

being a contour. Then a threshold is selected and contours above the threshold are considered contours and others are not. The method then takes a predicted contour and matches it to each of the 5 human contour maps, if the contour is not present in any of the 5 maps, then it is considered a false positive. The predicted contour map achieves maximum recall if it explains **all** of the human annotated contour maps.

We also later present results for the F-score (= $\frac{precision \times recall}{precision + recall}$) for OIS (optimal image scale), where we choose the optimal threshold per image. ODS (optimal dataset scale) where the threshold is chosen for the whole dataset. We also present the R50 score, which is the recall at 0.5 precision and AP, which is the area under the curve of the precision recall curve discussed earlier.

Comment on precision-recall evaluation. There are some discussions one could have on the validity of the precision-recall framework for measuring the performance of contours. Firstly, the BSDS dataset contains not only semantic contours - some users draw out also textures. Thus the evaluation method does not lead itself to semantic contours perfectly.

Furthermore for recall we expect the contours to match *each* human drawn map. This is not the best way to measure the contour map quality, as we know segmentation can be up to the user, and a self-supervised segmentation is "good" if it explains any single one of the segmentations. This can be further supported by measuring the human performance using this framework. Human drawn contours achieve an F-score of only 0.8 on the Berkeley Segmentation Dataset as reported in HED [XT15]. The ambiguity of BSDS can be seen in Figure 3.4.

However an argument can also be made in favour of measuring the recall as it is. One could appeal to the usefulness of a contour map, a self-supervised segmentation is useful if a user can easily interact with it. And that means he can recover all the possible segmentations from it and hence also all possible contours.

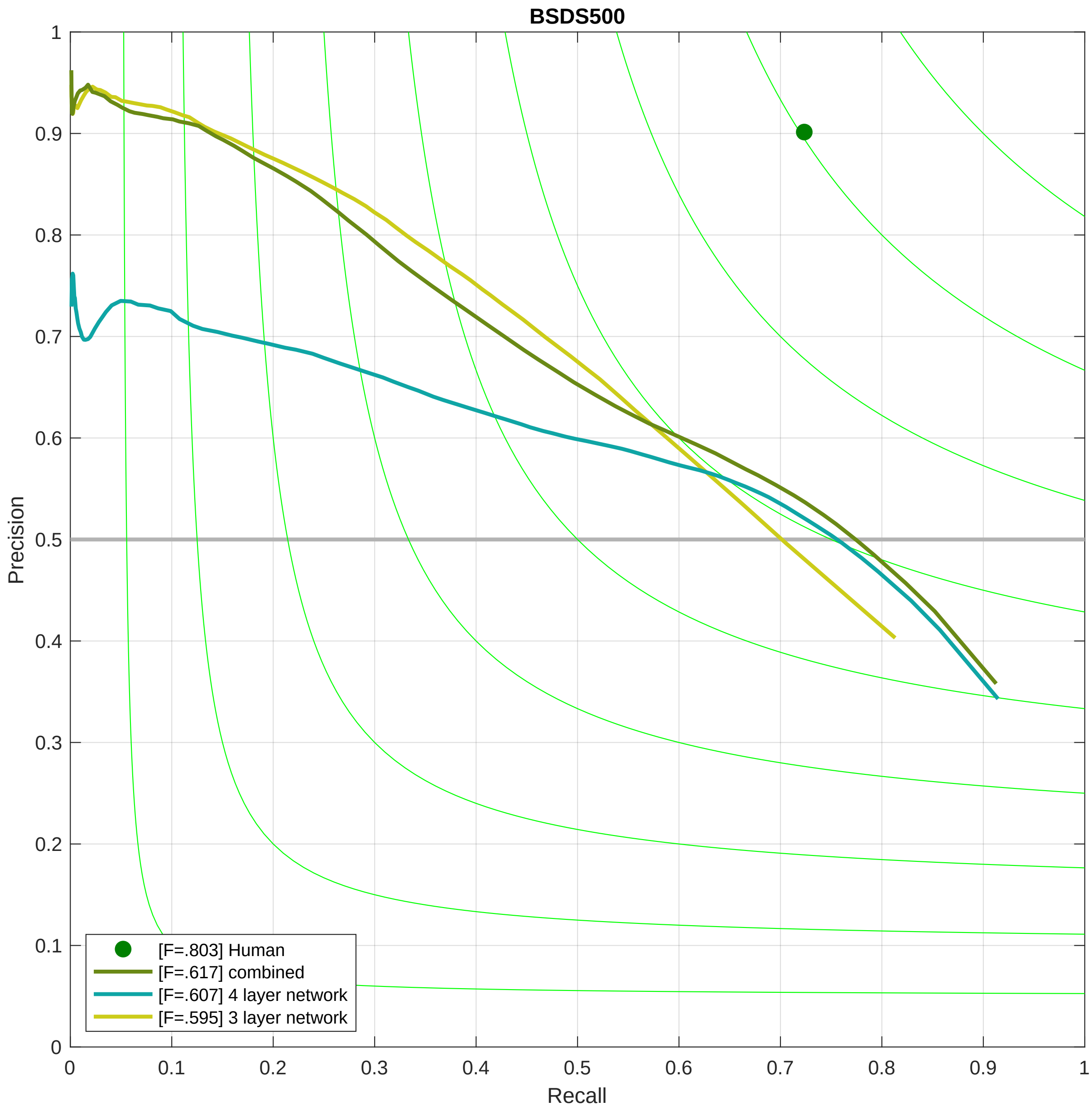


Figure 3.5: Comparison plot for different number of layers in the base network and the combined edges from both using an arithmetic mean.

3.4.1 Aggregating multiple scales

As we can see from Figure 3.5 the 3 layer encoder-decoder network generally produces a higher precision value i.e. the most of the contours it produces are contained in groundtruth contours, however it generally has a lower recall i.e. there are many groundtruth contours it does not produce. As compared to the the 4 layer network, which generally gives higher recall of contours but lower precision. This is not generally because the contours present in the 3 layer network are not in the 4 layer network, but rather it has to do with the threshold. For the precision recall curves we set a threshold and declare all contours which have a strength above that threshold to be positives. The 4 layer network does not favour contours that come up in the 3 layer network predictions (strength wise), even though that is the desired behaviour.

This behaviour is desired due to simple scale analysis, a 3 layer network generally gives less contours but those which are present should have a higher strength, since they are probable to be correct.

This is also why in Figure 3.5 we also plot out the results of the combination of the two scales i.e. one in which we take a simple arithmetic mean between the two predictions.

The results indicate that this combination gets the best of both worlds, a high precision but also good recall.

3.4.2 Comparison with other methods

As we can see we perform better than most unsupervised methods. The only method which achieves a better F-score, is Normalized Cuts [SM00], however we can see that still for many threshold our contours attain much higher precision.

This can be further seen in Figure 3.7, where of all the methods without supervision we achieve the highest Area Under the Curve (Average precision - AP). In fact out Area under

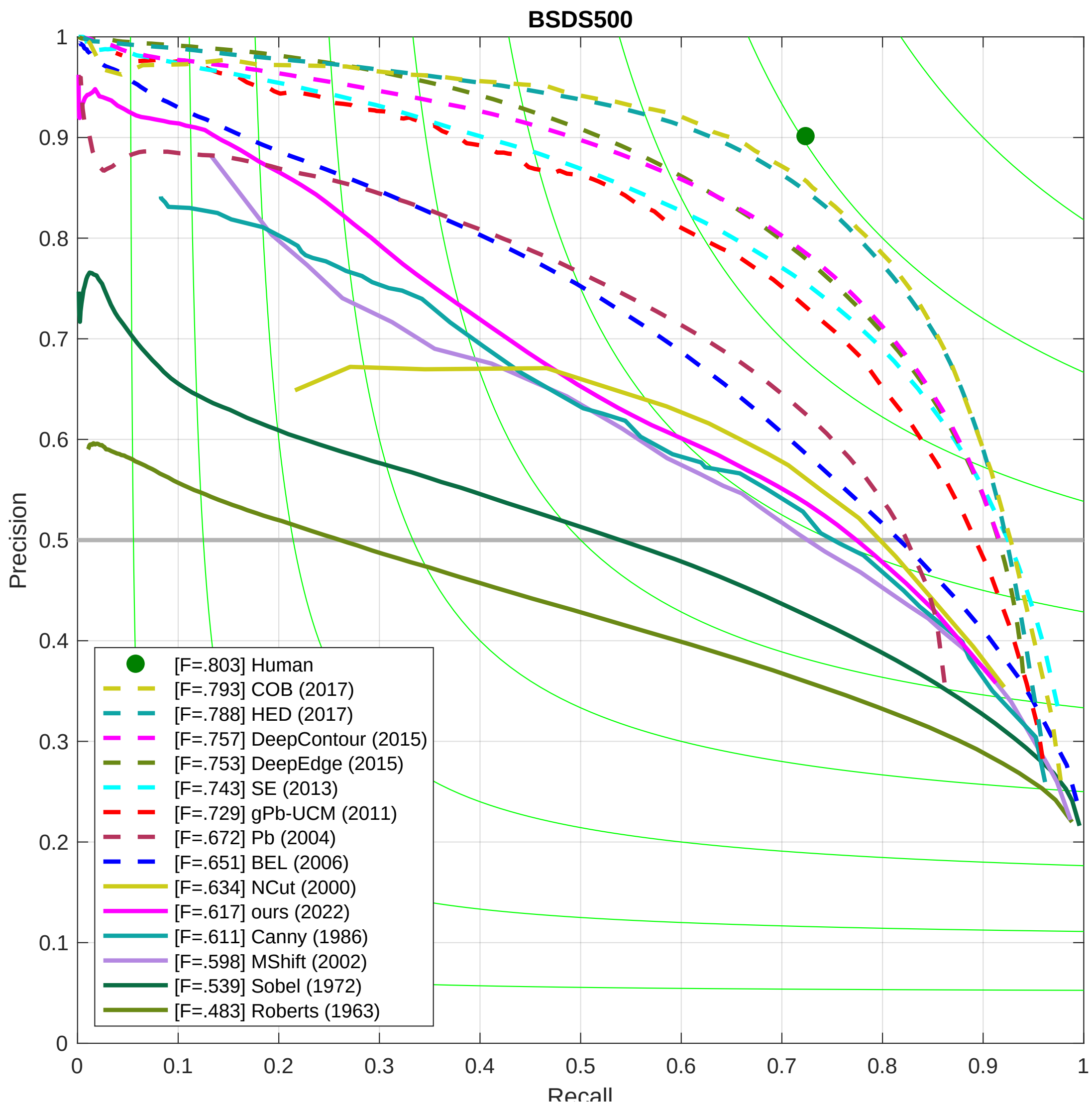


Figure 3.6: Comparison plot on the BSDS dataset for all methods. In continuous line are the unsupervised methods and dotted ones are the supervised methods.

the curve is nearing some supervised methods (like Pb). We can generally see in other metrics we are at the higher end of the measure, however we mainly suffer due to our poor recall. We do not always get all of the required contours in all the contour maps.

Method	ODS	OIS	AP	R50
Roberts	0.48	0.51	0.41	0.26
Sobel	0.54	0.58	0.50	0.54
Mean Shift	0.60	0.65	0.50	0.73
Canny	0.61	**0.68**	0.52	0.75
NCut	**0.63**	0.66	0.42	**0.80**
Ours	0.62	0.65	**0.63**	0.77
BEL	0.65	0.67	0.70	0.82
Pb	0.67	0.70	0.65	0.82
gPb	0.73	0.76	0.75	0.89
SE	0.74	0.76	0.80	**0.93**
HED	**0.79**	0.81	0.84	0.92
COB	**0.79**	**0.82**	**0.85**	**0.93**

Figure 3.7: Table comparing the optimal F-score for the OIS (optimal image scale) and ODS (optimal dataset scale), AP (averge precision) also known as AUC of precision recall curve, R50 (recall at 0.5 precision). All results below the dashed line are supervised.

It is quite impressive that even though many of these methods are specifically hand designed to get accurate contours, we are able to perform as well as them, despite optimizing a quite different objective.

Our objective just maximizes for agreement between the different segmentation heads and it is able to find semantically consistent contours.

In Figure 3.8 we can see examples of contours from our method, gPb [AMFM11] - a supervised method and groundtruth contours from one of the annotator.

As we can see the problem is difficult due to the sole nature of segmentation, due to ambiguity it is not known which contours might be required.

Even though labelling the reflection of the swan would usually seem unnecessary it is

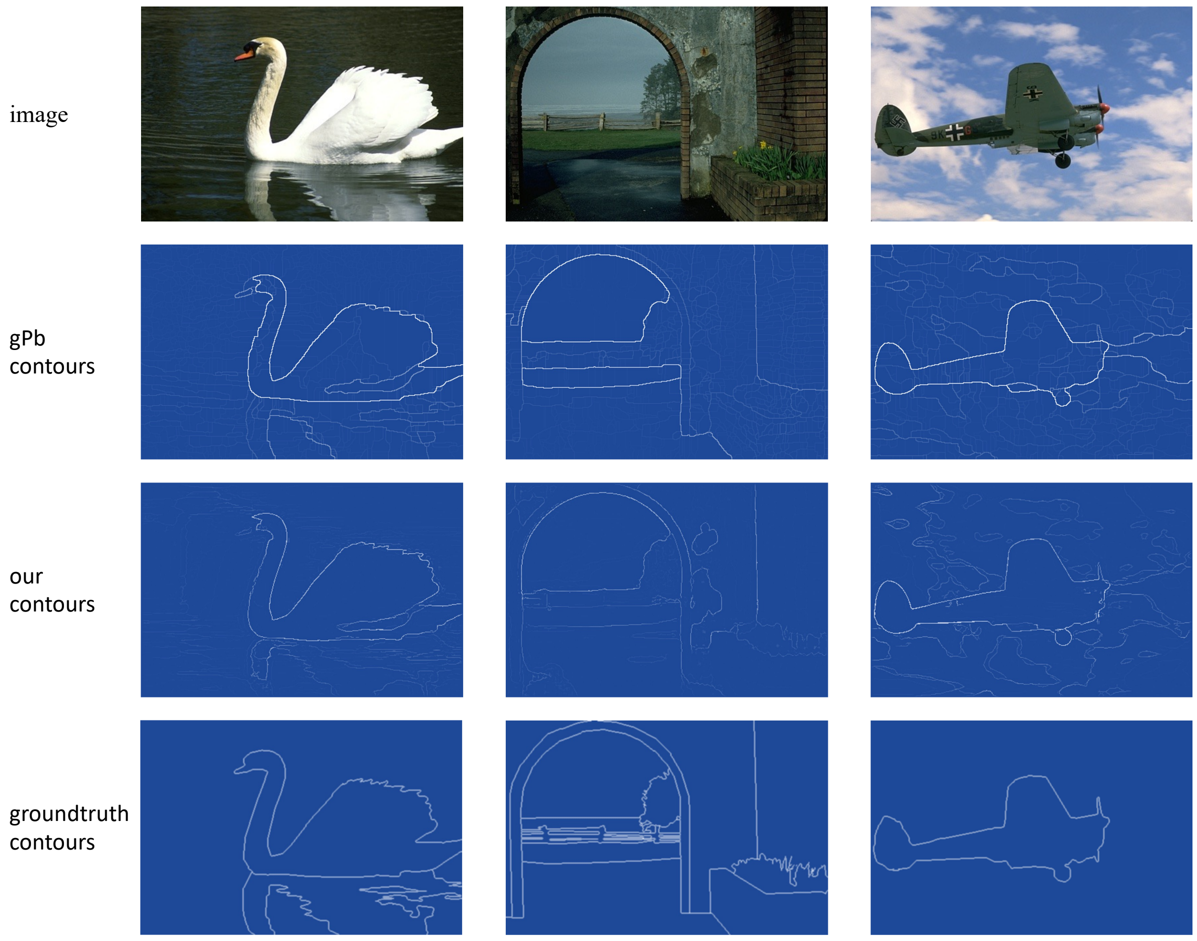


Figure 3.8: Images and contours produced by our method and gPb [AMFM11], which is supervised. The groundtruth image is just one of the groundtruth images of 5 given in the dataset. All 5 groundtruth contours can be found in the appendix.

done here by the annotator. Nonetheless our contours are very much comparable to ones produced by the supervised method in the presented images. However it must be said that image contrast does play a role for us as can be seen by the missing beak in our method.

3.4.3 Segmentation

We can also study our segmentation method, or more specifically our method to go from contours to a segmentation. Results for the 4 layer network can be found in Figure 3.10.

We firstly compare qualitatively to DFC [KKT20]. As we can see our results here are quite noisy, this we will show later is caused by the centring, however what we do better at is higher

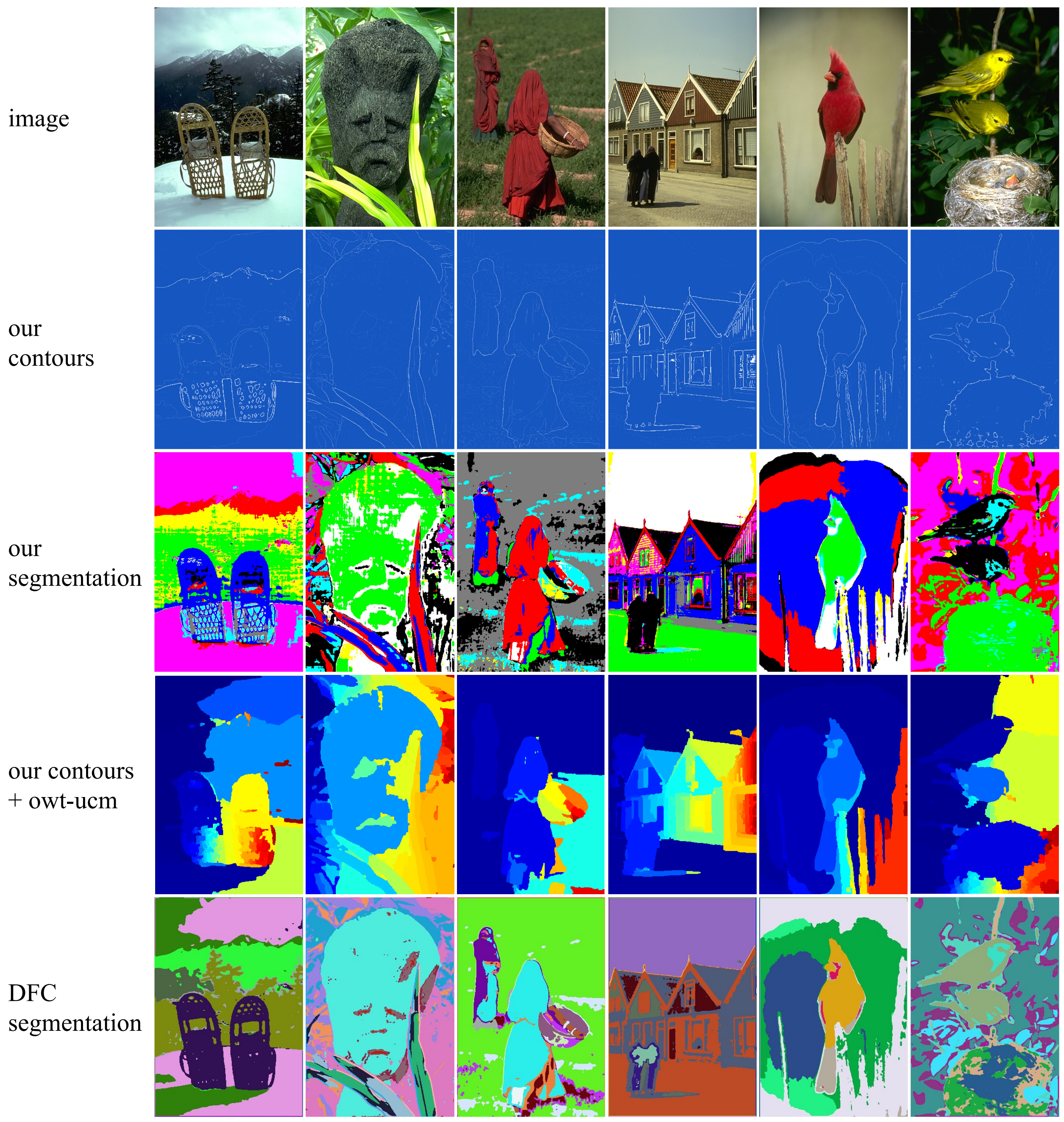


Figure 3.9: Segmentation comparison for the 4 layer network. Second row: contours produced by our method with non-maximal suppression (NMS was not used for the segmentation), third row: our segmentation from our contours (without NMS), fourth row: segmentation using owt-ucm [AMFM11] with our contours (without NMS), fifth row: segmentation from DFC [KKT20].

frequency segmentation. This is especially visible in the first image, where we are able to perfectly get all the small details in the sleigh. This is because of our network architecture, DCF uses a network which has the image as its input, which means that some of the high frequency details are lost during the processing, this is further exacerbated by their spatial continuity loss, which encourages nearby pixels to have the same label. Both methods follow the image edges very well and are able to label across image parts, like for the two sleighs.

We also compare our method to go from contours to segmentation with owt-ucm [AMFM11]. Owt-ucm takes the contour images and transforms it into a segmentation by first calculating the oriented watershed transform (owt), what the algorithm does is ensure piecewise edge strength continuity (similar to hysteresis in Canny [Can86]). Then thresholds are applied on the edgemap to get multiple (hierarchical) contours called the ultrametric contour map. This is simply transformed into a segmentation by filling in closed regions. The results we report are for a single threshold chosen to be 0.1. As we can see owt-ucm is not able to join disparate regions (like for the sleighs). Due to our method using again the image features to perform the segmentation, we are aware of things like texture etc, which allows us to be able to join separate region into the same class. However this also means we are slightly biased to colour and can sometimes classify things together incorrectly, like the windows and ladies on the picture of the houses.

Centring and noisy segmentation. Some of the noise in our segmentation can be attributed to the artifact of the upsampling network. This is a known issue in CNNs and was even covered in a Distill article [ODO16]. Most sources recommend to use Nearest Neighbour upsamplers to get rid of checkerboard patters (this can be seen in the mountains on the sleights image Figure 3.10). We do use these, however the issue still persists. Using a network without upsampling/downsampling solves the issue, however causes much computational overhead.

A second thing causing this issue is the centring. Centring (or batch normalization) has been used on logits before softmax in many sources seeking to avoid mode collapse [CTM+21],

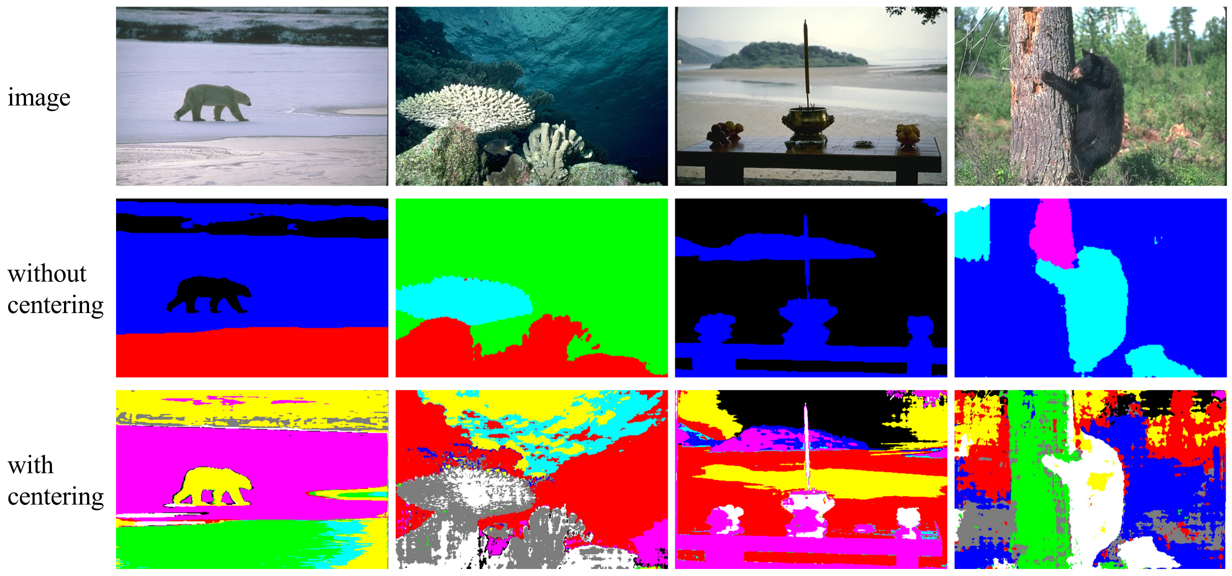


Figure 3.10: Segmentation comparison for the 4 layer network. Second row: contours produced by our method with non-maximal suppression (NMS was not used for the segmentation), third row: our segmentation from our contours (without NMS), fourth row: segmentation using owt-ucm [AMFM11] with our contours (without NMS), fifth row: segmentation from DFC [KKT20].

[KKT20]. However this is not a fool proof way of ensuring having many classes and can lead to noisy segmentations as we can see. The results without the centring are presented below. We can see they are much less noisy, however they do collapse to but a few classes, which is of course undesirable at the end of the day.

3.5 Conclusions

We can see we have produced a very unique self-distillation based setup for contour detection and segmentation. The method we have produced performs on par with other self-supervised methods for contour detection, even though we never specifically maximize for that objective. The results from our segmentation are also comparable to other state of the art self-supervised segmentation methods.

There are still however many elements and improvements that could be incorporated into our work.

Multi-scale. What is interesting is that scale is a very natural element for us and can be very easily incorporated into our system. This is the next step for our work, to have one network which is able to produce results at different scales which can all be combined. This is akin to what we are doing now however with different networks. It would be more efficient if we could get the same results running just a single network. An idea could be to incroporate the multi-scale neural fields used in Bacon [LVPW21] to get fields at multiple scales at ones, with just supervision at one level.

Segmentation noise. Another important avenue for us to explore would be to get segmentation without noise. An idea for how we could do this is to remove the use of centring and just rely on contours. What could we do with this setup? Currently edge strengths are small, so even though a contour would have a probability of 0.5, to our segmentation head this might not be enough to separate pixels into different classes. So perhaps an idea could be to increase the strength of contours - then we could have a non collapsing segmentation without centring, while also removing the noise.

Object recognition. This is not obvious how to do but a natural extension to segmentation would be object recognition. There have been many works trying to extend neural fields across scenes using various approaches like meta-learning [SMB+20], [TMW+21] or auto-decoding [PFS+19]. However these currently do not work quite well and are expensive to train. However perhaps in the future this generalization would also allow us to do recognition apart from just segmentation.

3D domain. Finally since all of the work here is been done with a Neural Field, we can extend our work to the 3D domain. Glossing over some details, our method would be able to segment a single scene in a self-supervised way while also producing contours for each keyframe.

Chapter 4

Conclusion

In this thesis, we have investigated the use of neural representations for self-supervised segmentation. In the first chapter we introduced SegDIP, a solution to the better defined problem of interactive image segmentation i.e. where a user is providing sparse supervision to the network.

We then introduced multiple extensions for our work. Firstly we used Neural Tangent Kernel literature to analyze the behaviour of our network. Secondly we introduced a way to generate images using the user provided semantics. Finally inspired by the Temporal Consistency loss we designed a method for self-supervised segmentation. The method was not successful in giving us the desired results (semantic looking self-supervised segmentation), however it did educate our choices for the following work.

In the second stage of our work, using the lessons learnt in SegDIP, we introduced a method for self-supervised segmentation. We learnt that an ensamble based method would be of use to us, and with the help of the ensamble we designed a unique self-distillation setup for image contour detection. We then designed a method to go from contours to segmentation. We compared our method to self-supervised contour detection methods.

4.0.1 Future work

3D. All of the work introduced here can be generalized to 3D. With the increasing use of Neural Fields in 3D Computer Vision [XTS+21], this system introduced especially in the second chapter would be of major use. Self-supervised segmentation is a budding topic, and doing these same methods in 3D are very difficult, since its unknown how exactly to represent this data. With the use of Neural Representations it has become simpler to train with just images and hence it is also simple to incorporate 2D learning ideas into these methods.

Improve performance of contour detection. Furthermore as previously mentioned there is still room to better the performance in our work, we are currently not using the full multi-scale capabilities of our convolutional network, which can be further explored.

We would also hope in the future the use of a reconstruction/generative objective can be explored for self-supervised object segmentation and recognition.

Discriminative vs generative. We hope our work is a step towards blending of generation and discrimination. These two tasks should not be separated in our models. A good segmentation model should inform our reconstruction and a good reconstruction should give us a better understanding of the scene. These two paradigms are currently separated in our methods, we either design generative networks to learn to model data or networks to segment and classify.

Increasingly many neuroscientists have been exploring these blended objectives in their work for example in GLOM by Hinton [Hin21] or in the Thousand Brain Theory book by Hawkins [HD21]. The idea is that our brains are constantly predicting future frames (generation) and this generation process should also help us better understand how to separately store different representations efficiently in our brain, thus a segmentation naturally arises.

Final objective of this work. Nonetheless we hope with this thesis we have shown that there is a lot of information to be learnt from a single image. Self-supervised segmentation from a single image can also be possible and should be further explored.

Bibliography

[AH77] K. Appel and W. Haken. Every planar map is four colorable. Part I: Discharging. *Illinois Journal of Mathematics*, 21(3):429 – 490, 1977.

[ALKF18] David Acuna, Huan Ling, Amlan Kar, and Sanja Fidler. Efficient interactive annotation of segmentation datasets with polygon-rnn++. 2018.

[AMFM11] Pablo Arbeláez, Michael Maire, Charless Fowlkes, and Jitendra Malik. Contour detection and hierarchical image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33(5):898–916, 2011.

[AOP$^+$18] Yağız Aksoy, Tae-Hyun Oh, Sylvain Paris, Marc Pollefeys, and Wojciech Matusik. Semantic soft segmentation. *ACM Trans. Graph. (Proc. SIGGRAPH)*, 37(4):72:1–72:13, 2018.

[BBI08] Shai Bagon, Oren Boiman, and Michal Irani. What is a good image segment? a unified approach to segment extraction. In *ECCV*, 2008.

[BI16] Yuval Bahat and Michal Irani. Blind dehazing using internal patch recurrence. In *2016 IEEE International Conference on Computational Photography (ICCP)*, pages 1–9, 2016.

[BJ88] P.J. Besl and R.C. Jain. Segmentation through variable-order surface fitting. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 10(2):167–192, 1988.

[BKC15] Vijay Badrinarayanan, Alex Kendall, and Roberto Cipolla. Segnet: A deep convolutional encoder-decoder architecture for image segmentation, 2015.

[BKSI20] Sefi Bell-Kligler, Assaf Shocher, and Michal Irani. Blind super-resolution kernel estimation using an internal-gan, 2020.

[BR08] Charles Bibby and Ian Reid. Robust real-time visual tracking using pixel-wise posteriors. In *Proceedings of the 10th European Conference on Computer Vision: Part II*, ECCV '08, page 831–844, Berlin, Heidelberg, 2008. Springer-Verlag.

[BST14] Gedas Bertasius, Jianbo Shi, and Lorenzo Torresani. Deepedge: A multi-scale bifurcated deep network for top-down contour detection, 2014.

[Can86] John Canny. A computational approach to edge detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-8(6):679–698, 1986.

[Cas16] Lluís Castrejón. Polyrnn : Polygon-based instance segmentation with recurrent neural networks. 2016.

[CGW21] Wuyang Chen, Xinyu Gong, and Zhangyang Wang. Neural architecture search on imagenet in four GPU hours: A theoretically inspired perspective. *CoRR*, abs/2102.11535, 2021.

[CKNH20] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey E. Hinton. A simple framework for contrastive learning of visual representations. *CoRR*, abs/2002.05709, 2020.

[CM02] D. Comaniciu and P. Meer. Mean shift: a robust approach toward feature space analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(5):603–619, 2002.

[CPSA17] Liang-Chieh Chen, George Papandreou, Florian Schroff, and Hartwig Adam. Rethinking atrous convolution for semantic image segmentation, 2017.

[CPW21] Lei Chu, Hao Pan, and Wenping Wang. Unsupervised shape completion via deep prior in the neural tangent kernel perspective. *CoRR*, abs/2104.09023, 2021.

[CRM00] D. Comaniciu, V. Ramesh, and P. Meer. Real-time tracking of non-rigid objects using mean shift. In *Proceedings IEEE Conference on Computer Vision and Pattern Recognition. CVPR 2000 (Cat. No.PR00662)*, volume 2, pages 142–149 vol.2, 2000.

[CTM+21] Mathilde Caron, Hugo Touvron, Ishan Misra, Hervé Jégou, Julien Mairal, Piotr Bojanowski, and Armand Joulin. Emerging properties in self-supervised vision transformers. *CoRR*, abs/2104.14294, 2021.

[DDS+09] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009.

[DGE15] Carl Doersch, Abhinav Gupta, and Alexei A. Efros. Unsupervised visual representation learning by context prediction. *CoRR*, abs/1505.05192, 2015.

[DTB06] P. Dollar, Zhuowen Tu, and S. Belongie. Supervised learning of edges and object boundaries. In *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*, volume 2, pages 1964–1971, 2006.

[DZ14] Piotr Dollár and C. Lawrence Zitnick. Fast edge detection using structured forests. *CoRR*, abs/1406.5549, 2014.

[EL99] A.A. Efros and T.K. Leung. Texture synthesis by non-parametric sampling. In *Proceedings of the Seventh IEEE International Conference on Computer Vision*, volume 2, pages 1033–1038 vol.2, 1999.

[FH04] Pedro F. Felzenszwalb and Daniel P. Huttenlocher. Efficient graph-based image segmentation. *International Journal of Computer Vision*, 59:167–181, 2004.

[FM81] K.S. Fu and J.K. Mui. A survey on image segmentation. *Pattern Recognition*, 13(1):3–16, 1981.

[GBI09] Daniel Glasner, Shai Bagon, and Michal Irani. Super-resolution from a single image. pages 349 – 356, 11 2009.

[GFS$^+$22] Niv Granot, Ben Feinstein, Assaf Shocher, Shai Bagon, and Michal Irani. Drop the gan: In defense of patches nearest neighbors as single image generative models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 13460–13469, June 2022.

[GG84] Stuart Geman and Donald Geman. Stochastic relaxation, gibbs distributions, and the bayesian restoration of images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-6(6):721–741, 1984.

[GSA$^+$20] Jean-Bastien Grill, Florian Strub, Florent Altché, Corentin Tallec, Pierre H. Richemond, Elena Buchatskaya, Carl Doersch, Bernardo Ávila Pires, Zhaohan Daniel Guo, Mohammad Gheshlaghi Azar, Bilal Piot, Koray Kavukcuoglu, Rémi Munos, and Michal Valko. Bootstrap your own latent: A new approach to self-supervised learning. *CoRR*, abs/2006.07733, 2020.

[GSI18] Yossi Gandelsman, Assaf Shocher, and Michal Irani. "double-dip": Unsupervised image decomposition via coupled deep-image-priors, 2018.

[Guz68] Adolfo Guzmán. Decomposition of a visual scene into three-dimensional bodies. In *Proceedings of the December 9-11, 1968, Fall Joint Computer Conference, Part I*, AFIPS '68 (Fall, part I), page 291–304, New York, NY, USA, 1968. Association for Computing Machinery.

[GW74a] J. N. Gupta and P. A. Wintz. Computer processing algorithm for locating boundaries in digital pictures. *Int. Joint Conf. Pattern Recognition*, 1974.

[GW74b] J. N. Gupta and P. A. Wintz. Multi-image modeling. *Technical Report TR-EE 74-24, School of Electrical Engineering, Purdue University*, 1974.

[HD75] R. Haralick and I. Dinstein. A spatial clustering procedure for multi-image data. *IEEE Transactions on Circuits and Systems*, 22(5):440–450, 1975.

[HD21] J. Hawkins and R. Dawkins. *A Thousand Brains: A New Theory of Intelligence*. Basic Books, 2021.

[HFW+20] Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. Momentum contrast for unsupervised visual representation learning. *CVPR*, 2020.

[Hin21] Geoffrey Hinton. How to represent part-whole hierarchies in a neural network, 2021.

[Hod61] Leo Hodes. machine processing of line drawings. 03 1961.

[HS12] Kaiming He and Jian Sun. Statistics of patch offsets for image completion. pages 16–29, 10 2012.

[HZH+22] Mark Hamilton, Zhoutong Zhang, Bharath Hariharan, Noah Snavely, and William T. Freeman. Unsupervised semantic segmentation by distilling feature correspondences. In *International Conference on Learning Representations*, 2022.

[IS15] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift, 2015.

[JGH18] Arthur Jacot, Franck Gabriel, and Clément Hongler. Neural tangent kernel: Convergence and generalization in neural networks. *CoRR*, abs/1806.07572, 2018.

[JMB76] Tenenbaum J. M and H. G. Barrow. Igs: a paradigm for integrating image segmentation and interpretation. *Int. Joint Conf. Pattern Recognition,*, 1976.

[KB14] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization, 2014.

[KKT20] Wonjik Kim, Asako Kanezaki, and Masayuki Tanaka. Unsupervised learning of image segmentation based on differentiable feature clustering. *IEEE Transactions on Image Processing*, 29:8055–8068, 2020.

[KSH12] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. Imagenet classification with deep convolutional neural networks. In *Proceedings of the 25th International*

Conference on Neural Information Processing Systems - Volume 1, NIPS'12, page 1097–1105, Red Hook, NY, USA, 2012. Curran Associates Inc.

[LBBH98] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[LDJ$^+$16] Di Lin, Jifeng Dai, Jiaya Jia, Kaiming He, and Jian Sun. Scribblesup: Scribble-supervised convolutional networks for semantic segmentation, 2016.

[LMS$^+$22] Xiankai Lu, Chao Ma, Jianbing Shen, Xiaokang Yang, Ian Reid, and Ming-Hsuan Yang. Deep object tracking with shrinkage loss. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44(5):2386–2401, 2022.

[LRAL07] Anat Levin, Alex Rav-Acha, and Dani Lischinski. Spectral matting. In *2007 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8, 2007.

[LSD15] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3431–3440, 2015.

[LVPW21] David B. Lindell, Dave Van Veen, Jeong Joon Park, and Gordon Wetzstein. BACON: band-limited coordinate networks for multiscale scene representation. *CoRR*, abs/2112.04645, 2021.

[LXOC22] Chenyang Lei, Yazhou Xing, Hao Ouyang, and Qifeng Chen. Deep video prior for video consistency and propagation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 1–1, 2022.

[Mar82] David Marr. *Vision: A Computational Investigation into the Human Representation and Processing of Visual Information*. Henry Holt and Co., Inc., New York, NY, USA, 1982.

[MBLS01] Jitendra Malik, Serge Belongie, Thomas Leung, and Jianbo Shi. Contour and texture analysis for image segmentation. *International Journal of Computer Vision*, 43:7–27, 06 2001.

[MFM04] D.R. Martin, C.C. Fowlkes, and J. Malik. Learning to detect natural image boundaries using local brightness, color, and texture cues. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(5):530–549, 2004.

[MFTM01] D. Martin, C. Fowlkes, D. Tal, and J. Malik. A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics. In *Proc. 8th Int'l Conf. Computer Vision*, volume 2, pages 416–423, July 2001.

[MKRLV22] Luke Melas-Kyriazi, Christian Rupprecht, Iro Laina, and Andrea Vedaldi. Deep spectral methods: A surprisingly strong baseline for unsupervised semantic segmentation and localization, 2022.

[MP71] Marvin Minsky and Seymour Papert. Progress report on artificial intelligence. 12 1971.

[MPTAG17] K.K. Maninis, J. Pont-Tuset, P. Arbeláez, and L. Van Gool. Convolutional oriented boundaries: From image segmentation to high-level tasks. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 2017.

[MR19] Indra Deep Mastan and Shanmuganathan Raman. Dcil: Deep contextual internal learning for image restoration and image retargeting, 2019.

[MST$^+$20] Ben Mildenhall, Pratul P. Srinivasan, Matthew Tancik, Jonathan T. Barron, Ravi Ramamoorthi, and Ren Ng. Nerf: Representing scenes as neural radiance fields for view synthesis. In *ECCV*, 2020.

[ODO16] Augustus Odena, Vincent Dumoulin, and Chris Olah. Deconvolution and checkerboard artifacts. *Distill*, 2016.

[OMS17] Chris Olah, Alexander Mordvintsev, and Ludwig Schubert. Feature visualization. *Distill*, 2017. https://distill.pub/2017/feature-visualization.

[PAED17] Deepak Pathak, Pulkit Agrawal, Alexei A. Efros, and Trevor Darrell. Curiosity-driven exploration by self-supervised prediction. *CoRR*, abs/1705.05363, 2017.

[Pan08] Caroline Pantofaru. *Studies in Using Image Segmentation to Improve Object Recognition*. PhD thesis, 01 2008.

[PFS+19] Jeong Joon Park, Peter Florence, Julian Straub, Richard A. Newcombe, and Steven Lovegrove. Deepsdf: Learning continuous signed distance functions for shape representation. *CoRR*, abs/1901.05103, 2019.

[RFB15] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation, 2015.

[RKB04] Carsten Rother, Vladimir Kolmogorov, and Andrew Blake. "grabcut": Interactive foreground extraction using iterated graph cuts. *ACM Trans. Graph.*, 23(3):309–314, aug 2004.

[RKK19] Sashank J. Reddi, Satyen Kale, and Sanjiv Kumar. On the convergence of adam and beyond, 2019.

[Rob63] Lawrence Roberts. *Machine Perception of Three-Dimensional Solids*. 01 1963.

[RSDM19] Tamar Rott Shaham, Tali Dekel, and Tomer Michaeli. Singan: Learning a generative model from a single natural image. In *Computer Vision (ICCV), IEEE International Conference on*, 2019.

[RZW+20] Dongwei Ren, Kai Zhang, Qilong Wang, Qinghua Hu, and Wangmeng Zuo. Neural blind deconvolution using deep priors, 2020.

[SCI17] Assaf Shocher, Nadav Cohen, and Michal Irani. "zero-shot" super-resolution using deep internal learning, 2017.

[SDR76] B. J. Schacter, L. S. Davis, and A. Rosenfeld. Scene segmentation by cluster detection in color spaces. *SIGART Bull.*, (58):16–17, jun 1976.

[SLOD21] Edgar Sucar, Shikun Liu, Joseph Ortiz, and Andrew J. Davison. imap: Implicit mapping and positioning in real-time. *CoRR*, abs/2103.12352, 2021.

[SM00] Jianbo Shi and J. Malik. Normalized cuts and image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(8):888–905, 2000.

[SMB+20] Vincent Sitzmann, Julien N.P. Martel, Alexander W. Bergman, David B. Lindell, and Gordon Wetzstein. Implicit neural representations with periodic activation functions. In *Proc. NeurIPS*, 2020.

[Sob14] Irwin Sobel. An isotropic 3x3 image gradient operator. *Presentation at Stanford A.I. Project 1968*, 02 2014.

[Suz20] Teppei Suzuki. Superpixel segmentation via convolutional neural networks with regularized information maximization. In *ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 2573–2577, 2020.

[SWL+21] Siyuan Shen, Zi Wang, Ping Liu, Zhengqing Pan, Ruiqian Li, Tian Gao, Shiying Li, and Jingyi Yu. Non-line-of-sight imaging via neural transient fields. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 43(7):2257–2268, jul 2021.

[SWRC06] Jamie Shotton, John Winn, Carsten Rother, and Antonio Criminisi. Textonboost: Joint appearance, shape and context modeling for multi-class object recognition and segmentation. pages 1–15, 07 2006.

[SWW+15] Wei Shen, Xinggang Wang, Yan Wang, Xiang Bai, and Zhijiang Zhang. Deepcontour: A deep convolutional feature learned by positive-sharing loss for contour detection. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3982–3991, 2015.

[TMW+21] Matthew Tancik, Ben Mildenhall, Terrance Wang, Divi Schmidt, Pratul P. Srinivasan, Jonathan T. Barron, and Ren Ng. Learned initializations for optimizing coordinate-based neural representations. In *CVPR*, 2021.

[TSM+20] Matthew Tancik, Pratul P. Srinivasan, Ben Mildenhall, Sara Fridovich-Keil, Nithin Raghavan, Utkarsh Singhal, Ravi Ramamoorthi, Jonathan T. Barron, and Ren Ng. Fourier features let networks learn high frequency functions in low dimensional domains. *CoRR*, abs/2006.10739, 2020.

[TTD21] Julian Tachella, Junqi Tang, and Mike Davies. The neural tangent link between cnn denoisers and non-local filters. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 8618–8627, June 2021.

[TV17] Antti Tarvainen and Harri Valpola. Weight-averaged consistency targets improve semi-supervised deep learning results. *CoRR*, abs/1703.01780, 2017.

[UVL20] Dmitry Ulyanov, Andrea Vedaldi, and Victor Lempitsky. Deep image prior. *International Journal of Computer Vision*, 128(7):1867–1888, Mar 2020.

[WNR74] J.S. Weszka, R.N. Nagel, and A. Rosenfeld. A threshold selection technique. *IEEE Transactions on Computers*, C-23(12):1322–1326, 1974.

[WSH$^+$22] Yangtao Wang, Xi Shen, Shell Hu, Yuan Yuan, James Crowley, and Dominique Vaufreydaz. Self-supervised transformers for unsupervised object discovery using normalized cut. 2022.

[XT15] Saining Xie and Zhuowen Tu. Holistically-nested edge detection. *CoRR*, abs/1504.06375, 2015.

[XTS$^+$21] Yiheng Xie, Towaki Takikawa, Shunsuke Saito, Or Litany, Shiqin Yan, Numair Khan, Federico Tombari, James Tompkin, Vincent Sitzmann, and Srinath Sridhar. Neural fields in visual computing and beyond. *CoRR*, abs/2111.11426, 2021.

[YFC$^+$20] Yingda Yin, Qingnan Fan, Dongdong Chen, Yujie Wang, Angelica Aviles-Rivero, Ruoteng Li, Carola-Bibiane Schnlieb, Dani Lischinski, and Baoquan Chen. Deep reflection prior, 2020.

Appendix

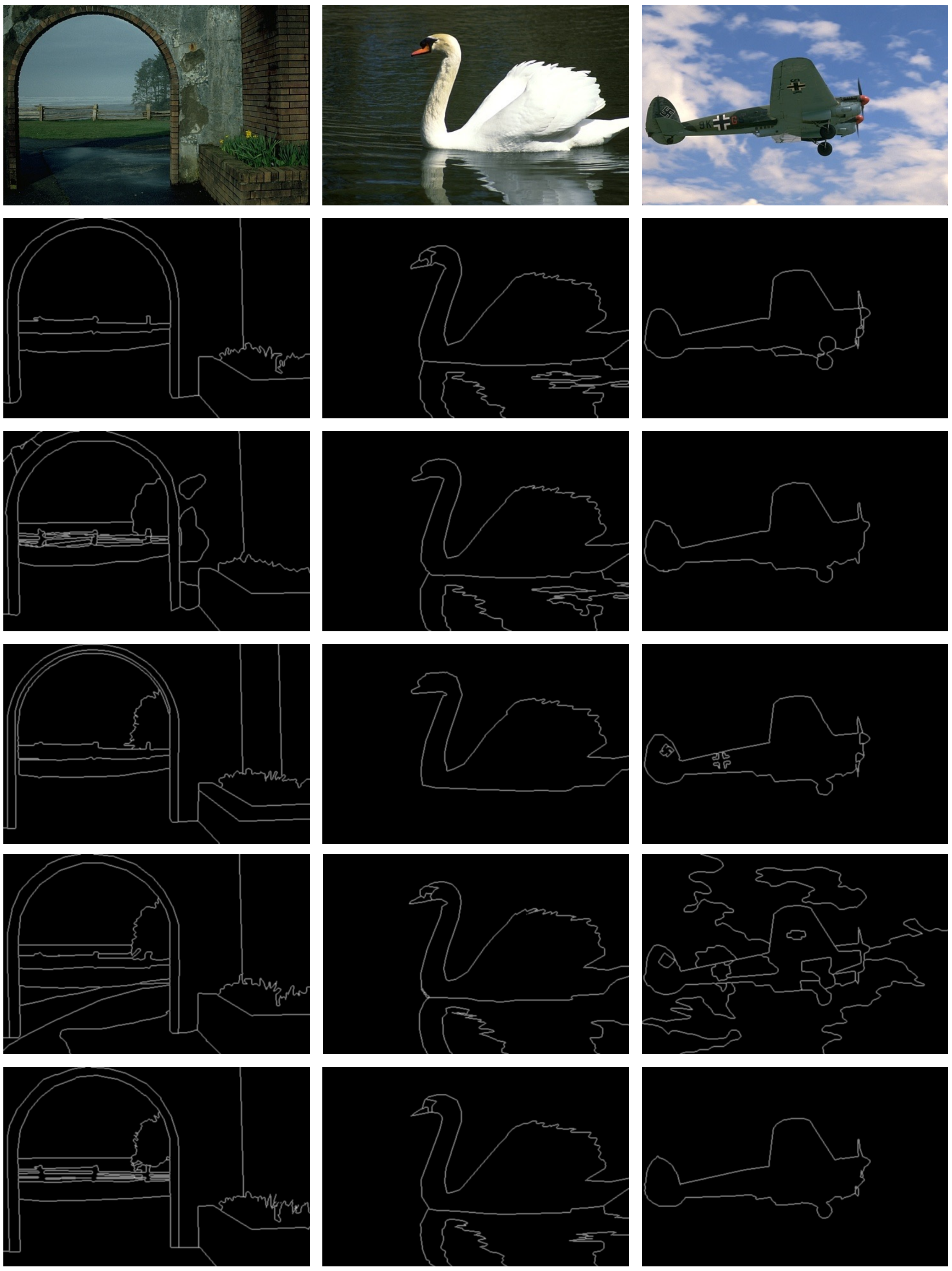


Figure 4.1: Groundtruth images and contours from the Berkeley Segmentation Dataset.

Ethical Considerations

Checklist

	Yes	No
Section 1: HUMAN EMBRYOS/FOETUSES		
Does your project involve Human Embryonic Stem Cells?		✓
Does your project involve the use of human embryos?		✓
Does your project involve the use of human foetal tissues / cells?		✓
Section 2: HUMANS		
Does your project involve human participants?		✓
Section 3: HUMAN CELLS / TISSUES		
Does your project involve human cells or tissues? (Other than from "Human Embryos/Foetuses" i.e. Section 1)?		✓
Section 4: PROTECTION OF PERSONAL DATA		
Does your project involve personal data collection and/or processing?		✓
Does it involve the collection and/or processing of sensitive personal data (e.g. health, sexual lifestyle, ethnicity, political opinion, religious or philosophical conviction)?		✓
Does it involve processing of genetic information?		✓
Does it involve tracking or observation of participants? It should be noted that this issue is not limited to surveillance or localization data. It also applies to Wan data such as IP address, MACs, cookies etc.		✓
Does your project involve further processing of previously collected personal data (secondary use)? For example Does your project involve merging existing data sets?		✓
Section 5: ANIMALS		
Does your project involve animals?		✓
Section 6: DEVELOPING COUNTRIES		
Does your project involve developing countries?		✓
If your project involves low and/or lower-middle income countries, are any benefit-sharing actions planned?		✓
Could the situation in the country put the individuals taking part in the project at risk?		✓
Section 7: ENVIRONMENTAL PROTECTION AND SAFETY		
Does your project involve the use of elements that may cause harm to the environment, animals or plants?		✓
Does your project deal with endangered fauna and/or flora /protected areas?		✓
Does your project involve the use of elements that may cause harm to humans, including project staff?		✓
Does your project involve other harmful materials or equipment, e.g. high-powered laser systems?		✓

	Yes	No
Section 8: DUAL USE		
Does your project have the potential for military applications?		✓
Does your project have an exclusive civilian application focus?		✓
Will your project use or produce goods or information that will require export licenses in accordance with legislation on dual use items?		✓
Does your project affect current standards in military ethics – e.g., global ban on weapons of mass destruction, issues of proportionality, discrimination of combatants and accountability in drone and autonomous robotics developments, incendiary or laser weapons?		✓
Section 9: MISUSE		
Does your project have the potential for malevolent/criminal/terrorist abuse?		✓
Does your project involve information on/or the use of biological-, chemical-, nuclear/radiological-security sensitive materials and explosives, and means of their delivery?		✓
Does your project involve the development of technologies or the creation of information that could have severe negative impacts on human rights standards (e.g. privacy, stigmatization, discrimination), if misapplied?		✓
Does your project have the potential for terrorist or criminal abuse e.g. infrastructural vulnerability studies, cybersecurity related project?		✓
SECTION 10: LEGAL ISSUES		
Will your project use or produce software for which there are copyright licensing implications?		✓
Will your project use or produce goods or information for which there are data protection, or other legal implications?		✓
SECTION 11: OTHER ETHICS ISSUES		
Are there any other ethics issues that should be taken into consideration?		✓

Discussion

Dangers of Semantic Segmentation. It is true that semantic segmentation can have major usage an impact in many different ethically controversial areas like face detection etc. However, the exact problem we are dealing with does not concern recognition, which reduces the scope for applications of our system to such areas.

Dangers of image generation. Some elements of this work include image generation. There are generally issues with generation concerning copyright. However in this work for generation we are not synthesizing images of quality similar to the target ones.

Datasets. The datasets used in this thesis, particularly the Berkeley Segmentation Dataset [MFTM01] are internet open license images collected and labeled by humans. The datasets are for non-commercial use and have been cited with the appropriate references. This thesis involves neither human, animal based data nor personal privacy data.

Concerns. If you have any additional concerns about the work, please contact the author.