FACULTY OF INFORMATION TECHNOLOGY AND ELECTRICAL ENGINEERING

Otso Suvilehto

WORKFLOW FOR REDUCING SEMANTIC SEGMENTATION ANNOTATION TIME

Master's Thesis
Degree Programme in Computer Science and Engineering
December 2020

Suvilehto O. (2020) Workflow for Reducing Semantic Segmentation Annotation Time. University of Oulu, Degree Programme in Computer Science and Engineering, 72 p.

ABSTRACT

Semantic segmentation is a challenging task within the field of pattern recognition from digital images. Current semantic segmentation methods that are based on neural networks show great promise in accurate pixel-level classification, but the methods seem to be limited at least to some extent by the availability of accurate training data. Semantic segmentation training data is typically curated by humans, but the task is rather slow and tedious even for humans. While humans are fast at checking whether a segmentation is accurate or not, creating segmentations is rather slow as the human visual system becomes limited by physical interfaces such as hand coordination for drawing segmentations by hand. This thesis evaluates a workflow that aims to reduce the need for drawing segmentations by hand to create an accurate set of training data.

A publicly available dataset is used as the starting-point for the annotation process, and four different evaluation sets are used to evaluate the introduced annotation workflow in labour efficiency and annotation accuracy.

Evaluation of the results indicates that the workflow can produce annotations that are comparable to manually corrected annotations in accuracy while requiring significantly less manual labour to produce annotations.

Keywords: semi-automatic, neural network, training data, dataset

Suvilehto O. (2020) Työnkulku semanttisen segmentoinnin annotointiajan vähentämiseen. Oulun yliopisto, Tietotekniikan tutkinto-ohjelma, 72 s.

TIIVISTELMÄ

Semanttinen segmentointi on haastava osa-alue hahmontunnistusta digitaalisista kuvista. Tämänhetkiset semanttiset segmentaatiomenetelmät, jotka perustuvat neuroverkkoihin, osoittavat suurta potentiaalia tarkassa pikselitason luokittelussa, mutta ovat ainakin osittain tarkan koulutusdatan saatavuuden rajoittamia. Semanttisen segmentaation koulutusdata on tyypillisesti täysin ihmisten annotoimaa, mutta segmentaatioiden annotointi on hidasta ja pitkäveteistä. Vaikka ihmiset ovat nopeita tarkistamaan ovatko annotaatiot tarkkoja, niiden luonti on hidasta, koska ihmisen visuaalisen järjestelmän nopeuden ja tarkkuuden rajoittavaksi tekijäksi lisätään fyysinen rajapinta, kuten silmä-käsi-koordinaatio piirtäessä segmentaatioita käsin. Tämä opinnäytetyö arvioi kokonaisvaltaisen semanttisten segmentaatioiden annotointitavan, joka pyrkii vähentämään käsin piirtämisen tarvetta tarkan koulutusdatan luomiseksi.

Julkisesti saatavilla olevaa datajoukkoa käytetään annotoinnin lähtökohtana, ja neljää erilaista evaluointijoukkoa käytetään esitetyn annotointitavan työtehokkuuden sekä annotaatiotarkkuuden arviointiin.

Evaluaatiotulokset osoittavat, että esitetty tapa kykenee tuottamaan annotaatioita jotka ovat yhtä tarkkoja kuin käsin korjatut annotaatiot samalla merkittävästi vähentäen käsin tehtävän työn määrää.

Avainsanat: puoli-automaattinen, neuroverkko, koulutusdata, datajoukko

TABLE OF CONTENTS

FOREWORD

What is learning? Certainly, a question that has riddled humanity for ages, and many agree that it is at least in part an accumulation of past experiences that result in some total of information that relates to these past experiences. This analogy is certainly true for current machine learning methods that iteratively converge towards a model of information that is defined by experiences – training samples – on each iteration. It is generally believed that the learned model cannot be more accurate than the training samples that were used to learn the model. This concept leads to the pursuit of more accurate and larger quantities of training samples while in pursuit of more accurate models on a set of information.

Human performance is still the gold standard in many tasks, including semantic segmentation. As such human labour is commonly used to create training samples for semantic segmentation, but the task may prove to be undesirable, repetitive, and tedious. Neural networks have proved to be effective at tasks that humans find trivial, and are very promising for increasingly complex tasks. Such methods are particularly interesting for increasing human efficiency via automation, so that humans can focus on more rewarding and more complex tasks, perhaps to push us towards an increasingly brilliant future.

The aim of this thesis is to evaluate whether neural networks can be used to help in the creation of semantic segmentation training samples. The results are evaluated by comparing training samples created by humans and training samples of which creation was assisted by a machine learning algorithm.

I thank Visidon Oy for the opportunity to work on and research this subject. As with learning in general accumulating, I hope that this work contributes even in the tiniest bit to helping humanity rid of tedious, undesirable tasks. I also thank my co-workers, friends and family for all the support, insight, and opinions they have shared with me during this process. And finally but not least significantly, a special thank you to the thesis supervisors Janne Heikkilä and Jari Hannuksela for guiding me through this milestone in life.

Oulu, December 8th, 2020

Otso Suvilehto

LIST OF ABBREVIATIONS AND SYMBOLS

GUI	graphical user interface
NCC	normalized cross-correlation
SVM	support vector machine
UML	unified modeling language
IoU	Intersection-over-Union
mIoU	mean Intersection-over-Union
MAC	multiply-accumulate
Web	Wold-Wide Web
TDaaS	Training-Data-as-a-Service
ReLU	Rectified linear unit
MSE	Mean-square error
XE	cross-entropy
SOTA	State-of-The-Art
KNN	K-Nearest neighbour
FBS	Fast Bilateral Solver
Pascal VOC	Pascal Visual Objects Challenge
COCO	Common Objects in Context
GPU	Graphics processing unit
FCN	Fully Convolutional Neural Network
ASPP	Atrous Spatial Pyramid Pooling

N	number of samples
r_{ij}	matrix element
s	distance
t	time
c	class

R	region
I	image
$\emptyset$	empty set
β	shape factor
ϵ_{k}	error signal at instant k
θ	parameter vector
σ^2	variance
α	alpha
$\hat{X}$	feature space X
$\hat{Y}$	feature space Y
$\longmapsto$	feature space mapping
$\varphi()$	activation function
$\exp()$	exponential function
$\arg()$	argument
$\max()$	maximum
$\cup$	union
$\cap$	intersection

1. INTRODUCTION

Computational image analysis is used widely in today's society in various applications ranging from mobile phones to cars, security systems, medical, and military applications. Many modern image analysis systems are based on machine learning algorithms. These systems rely heavily on training data curated by humans to complete the learning process. Image data that demonstrates a problem and the solution to that problem is fed to machine learning algorithms that fit a number of parameters to the data in an attempt to capture a generalised model of the problem at hand. These models are then saved and shipped across the world in various intelligent image processing and camera applications. But methods that are based on machine learning need good training data that describes the problem and solution accurately and broadly. This need for data has opened up a market for data collection and annotation. Training data is usually curated by humans to ensure sufficiently high quality, but manual labour is costly [1, 2].

Modern machine learning algorithms can benefit from enormous quantities of annotated data, requiring potentially a lot of manual work. A typical commercial image processing application that is based on deep learning utilises an image dataset that contains from tens of thousands to up to hundreds of thousands of annotated images.

A particularly expensive type of data annotation is semantic segmentation, where the areas of objects and regions such as the sky, trees, or humans are drawn, often by hand, to construct image - semantic segmentation mask pairs to be fed to machine learning algorithms. When trained with such data, these algorithms can then be employed to very accurately localise objects and regions in an image at pixel-level. While some semantic segmentation datasets are available publicly and free to use for academic work, teams and companies around the world are finding themselves needing different, more accurate and larger quantities of training data than is available publicly. This need for more and more training data has opened up a market for tools and services that specialise in image annotation for use in machine learning processes.

This research focuses on evaluating a specific process that aims to reduce the amount of manual work needed to annotate human outlines in images, enabling annotators to annotate more images for the same amount of work. The aim is to gain more insight into a specific to some extent self-feeding workflow on semi-automatic semantic segmentation which has been mentioned very briefly in the literature on semantic segmentation solutions. This includes summarising the average annotation time of popular publicly available semantic segmentation datasets and analysing the annotation speed and accuracy of a portrait segmentation dataset using one specified semi-automatic annotation procedure. The chosen data generation workflow will be evaluated on whether it can satisfy reasonable annotation accuracy requirements while reducing the amount of manual labour required to complete a dataset.

Obvious use-cases for this type of image data are in the development of image stylisation applications, such as mobile phone camera applications or photo editors, while the use-cases for the topic of this research would be in software and services that specialise in producing this type of image data.

2. DEEP NEURAL NETWORKS

To understand the need for large quantities of annotated training data, a basic understanding of machine learning and deep learning concepts and their role in various fields of technology is required. This work focuses on training data for deep learning and utilises deep learning methods as a part of the annotation process. As such the focus in this chapter is on neural networks and more specifically deep learning. Deep learning is an umbrella subset of machine learning methods that leverage large, deep networks of artificial neurons.

Mitchell defines machine learning in his work Machine Learning (1997) as "A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E" [3].

2.1. Neural Networks

Artificial neural networks is a specific subset of machine learning with traces all the way back to Threshold Logic Unit [4] introduced by McCulloch in 1943 and Rosenblatt's perceptron [5] first introduced in 1957. Both of these works are loosely inspired by biological neurons to represent information. Similar to biological neurons, their functionality is based on the concept that the activation or output of a neuron depends on the received input of the neuron. Theoretically, a network of such activation units could represent any arbitrary function or a model, which makes them a particularly interesting basis for applications that convert information such an image to another format such as a word, or a value that denotes a specific word. It has been shown that neural networks are universal function approximators [6, 7].

In deep learning, typically the function that performs information conversion — from here on called feature space mapping — is learned through parameter optimisation on given training data. Mathematically the use-case of neural networks is to approximate some function $f(x)$ which can be denoted as a feature space mapping $\hat{f}_x : \hat{X} \longmapsto \hat{Y}$ from some input feature space $\hat{X}$ to some target feature space $\hat{Y}$. In the use-case of this thesis $\hat{f}_x$ performs mapping from colour image feature space into a binary semantic segmentation image. To learn this feature space mapping, a large quantity of $\hat{X} \longmapsto \hat{Y}$ sample pairs is needed.

2.1.1. Neurons and Forward-Propagation

Computationally, the output y_k of a given neuron k for an input $\hat{x}$ of size N is defined as

$$y_k = f(\hat{x}) = \varphi(b_k + \sum_{i=0}^{N} w_{ki} x_i) \tag{1}$$

where w_{ki} is the weight of neuron k for each input x_i, b_k is bias and φ is an activation function such as a threshold or a sigmoid function. Weights $\hat{w}_k$ and biases b_k are

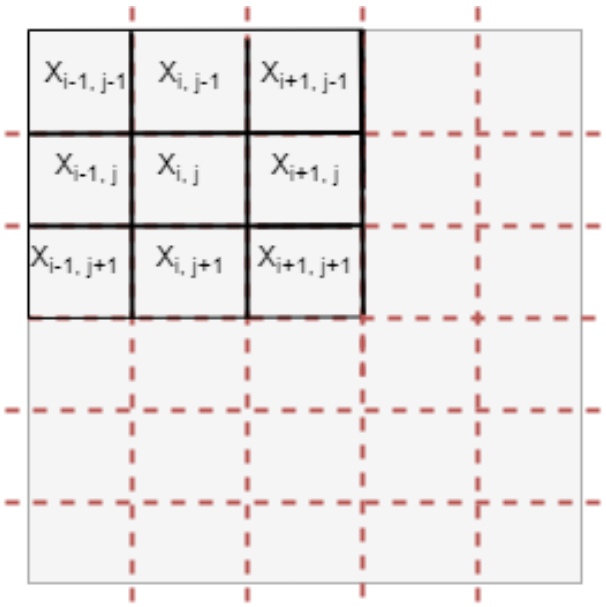


Figure 1. 3x3 convolutional kernel w at coordinate $X_{i,j}$.

the main adjustable parameters of a neural network. Equation (1) can be extended to convolutional neuron with a weight kernel $\hat{w}_k$ of size $M * N$ so that

$$y_k = f(\hat{x}_{i,j}) = \varphi(b_k + \sum_m \sum_n w_{i-m,j-n} x_{i,j}), \tag{2}$$

Convolutional neuron is typically used for array inputs such as images when the number of elements is larger than 1. Figure 1 visualises a 3-by-3 convolutional kernel around input $X_{i,j}$. In practice, many deep learning frameworks implement convolution as cross-correlation and call it convolution.

Each neuron can be thought of as a decision boundary where weights determine the direction of the boundary in feature space and bias defines the position. To solve nonlinear feature spaces, an activation function is used to introduce nonlinearity into the system. In multi-layer neural networks, the output of an activation function φ is used as input to the next neuron in the network. This defines the forward propagation of a neural network. Figure 2 shows a simple neural network with two inputs X_i, one hidden layer with four neurons y_k and one final output neuron Y. Neural networks typically have many consecutive layers of neurons. The number of layers is commonly referred to as the depth of a neural network. This depth can be described as a chain so that a neural network $f^*(x)$ is a chain of layers of neurons defined as

$$f^*(x) = f^n(f^{n-1}(,,,(f^2(f^1(x))))) \tag{3}$$

where $f^n(x)$ is the output of the nth layer in the chain of operations. This notion of a neural network being a chain is important later when discussing back-propagation, the basis for learning in neural networks.

The activation function φ is a continuous function with a well-defined derivate along the real x axis. For $f_x : \hat{X} \longmapsto \hat{Y}$ to be able to represent a non-linear mapping, a nonlinearity needs to be introduced into the network. Otherwise the output y_k would be a linear transformation of the input $\hat{x}$. This nonlinearity can be achieved by

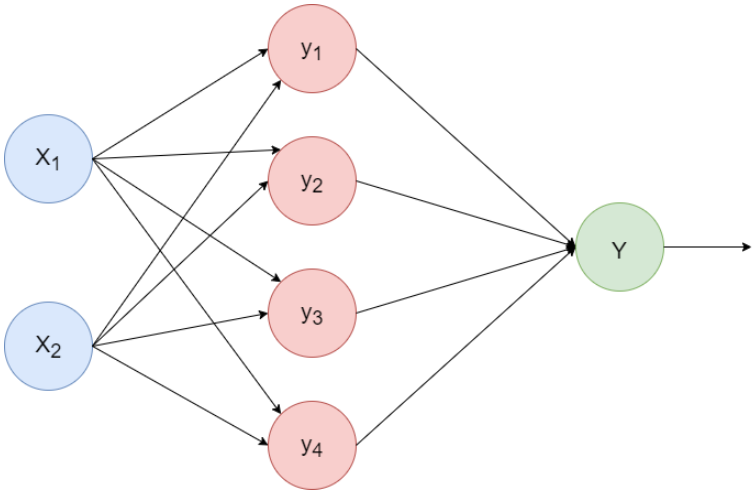


Figure 2. A simple neural network with one hidden layer.

selecting a nonlinear activation function φ [8]. Figure 3 shows various commonly used activation functions. From the activation functions presented in Figure 3, Rectified Linear Unit and its variations are recently most popular, though sigmoid and tanh have been popular in the past. The earlier work in Threshold Logic Units used a threshold activation function.

2.2. The Learning Process of Neural Networks

Neural networks' ability to learn is based on three main parts — an optimization algorithm, back-propagation and a loss function. Loss function, often also referred to as cost function, is a function that simply calculates the error of output y' in comparison to the ground-truth label y. Two common loss functions include mean-square error (MSE) and cross-entropy (XE) loss function.

$$MSE = \frac{1}{N}\sum_{i=1}^{N}(y - y')^2 \tag{4}$$

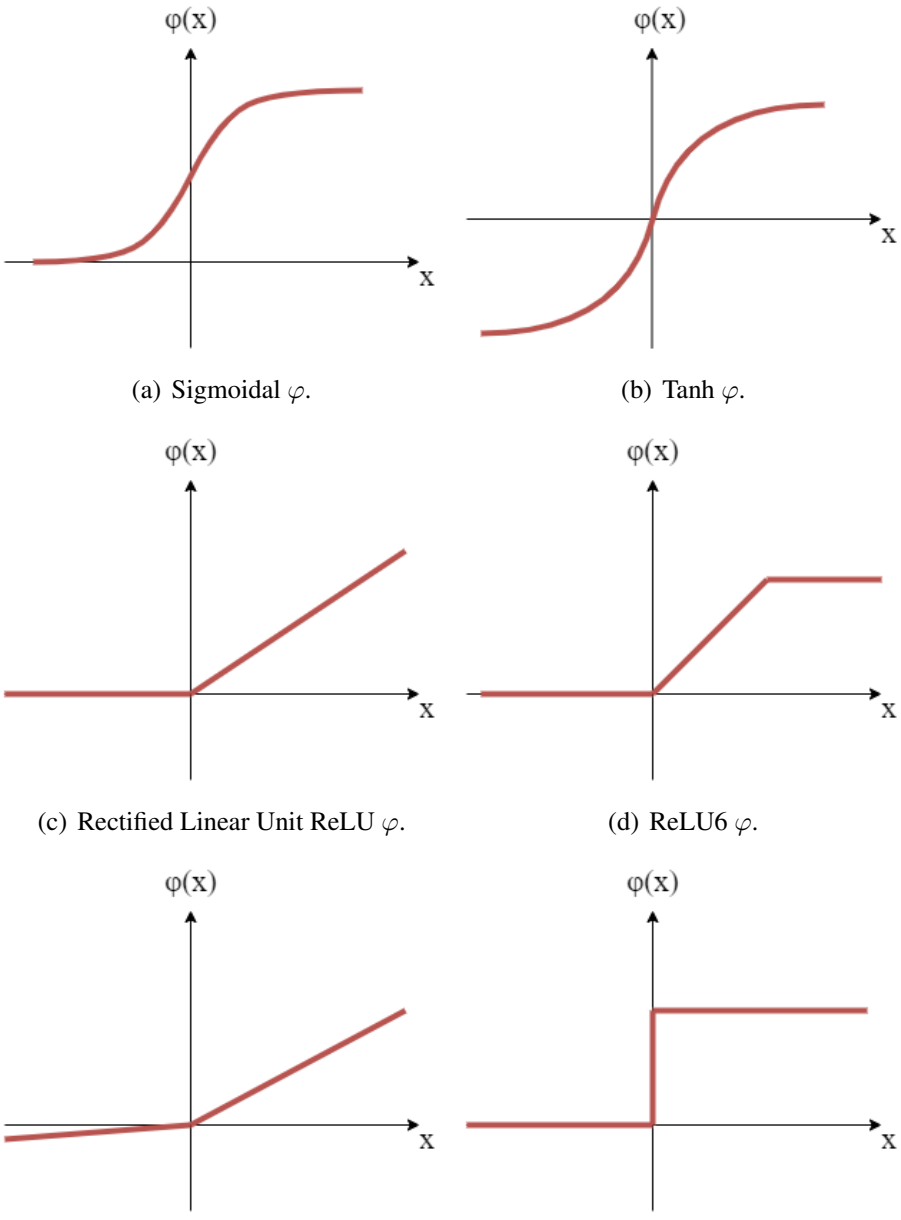
$$XE = -\sum_{c=1}^{M}(y_c log(p_{y'_c})) \tag{5}$$

Cross-entropy loss function

Equation (5) defines the cross-entropy loss function where M is the number of classes c and y_c is the ground-truth output for class c. log is the natural logarithm and $p_{y'_c}$ is the output of the neural network where the output y' is represented as a probability that the output for class c is y. Cross-entropy is commonly used for classification networks, and this thesis also uses cross-entropy to classify pixels in the training process. The result of the loss function is propagated through the network, and an optimisation algorithm adjusts each trainable parameter in the network.

Back-propagation

Back-propagation, or more specifically the reverse accumulation of automatic differentiation creates the basis for efficiently propagating the error gradient through the network and enabling optimisation algorithms to fine-tune each parameter. Back-propagation is based on the chain-rule of differentiation which leads to neural networks requiring operations that have well-defined derivates for propagating the error of output through the network. The propagated error is used to calculate changes in all network parameters to minimise the output error for every input-output-target pair. Back-propagation's modern form—the reverse accumulation mode of automatic differentiation—was first introduced by Linnainmaa in 1970 [9, 10, 11]. The term



(a) Sigmoidal φ. (b) Tanh φ.

(c) Rectified Linear Unit ReLU φ. (d) ReLU6 φ.

(e) Leaky ReLU φ. (f) Threshold φ.

Figure 3. Different activation functions.

back-propagation and its use in neural networks was first introduced in [12] and later popularised in [13].

Parameter optimization

Parameter optimisation is usually done iteratively with an optimisation algorithm that is based on gradient descent. Each parameter is adjusted in small increments to get the error value towards 0, to a very small number, along the error gradient in the opposite direction [8 p. 173] [14]. Gradient descent was originally introduced by Cauchy in 1847 [8 p. 81].

2.3. Building-Blocks

In practice, neural networks can be described as a construction of various building-block operations such as convolutional, fully-connected, pooling, upsampling and softmax operations. Each of them can be used to achieve different end-goals, and a brief description of the most common building-blocks is in place to better understand the structure of deep neural networks which tend to utilise more than just convolutions and activation functions. This chapter briefly explains some of the key building-blocks which are relevant to semantic segmentation and are also used in this work.

Pooling layer

Pooling layers perform local or global dimension reduction with methods such as averaging or taking the maximum value in a specified pooling window. Global pooling layers perform dimension reduction from an input array to a scalar value so that a single output value is selected from the pooling window with the pooling window size being the size of the input. Local pooling layers operate in specified window sizes such as 2-by-2 input neurons and output the average or maximum value in the window, effectively reducing the number of elements by a factor of 4.

Upsampling layers

Upsampling layers serve a purpose that is the opposite of pooling layers and are typically used in convolutional neural networks when increasing the layer size is required. Some common methods for upsampling are bi-linear, bi-cubic, and nearest-neighbour interpolations as well as transpose convolution which effectively learns the upsampling kernel.

Transpose convolution is usually implemented in machine learning frameworks by switching the forward- and backpropagation passes and it is often referred to as fractionally strided convolution or deconvolution, although as a name deconvolution is inaccurate [15 p. 359].

Skip-connection

Skip-connection is a widely used architectural element that passes information from a layer to a subsequent layer, skipping a number of layers between the two connected

layers. Skip-connections were proposed for deep convolutional neural networks by Long et al. in [16, 17]. The reasoning behind using skip-connections is to preserve and propagate information to subsequent parts of a neural network.

Batch normalisation

Batch normalisation helps speed up training by normalising layer inputs for each mini-batch. Introduced by Ioffe and Szegedy in 2015, the authors reported matching accuracy with 14 times fewer steps in training, and exceeding original results by a significant margin [18]. Batch normalisation addresses a challenge in training large networks with stochastic gradient descent caused by changes in network parameters affecting the inputs to all subsequent layers. This parameter — input relationship is challenging because the parameters need to adjust to changes in the input distributions. Batch normalisation addresses this issue by normalising the means and variances of each input to a layer. Normalising the inputs also reduces the back-propagated gradients' dependency on the parameter scales allowing the use of larger learning rates as the risk of vanishing gradients is reduced.

Fully-connected layer

A fully-connected layer effectively performs regression via matrix multiplication in which all outputs of the previous layer are fully connected to all neurons in the fully-connected layer to output a single vector of values. Fully-connected layer is typically used to perform regression at the end of a neural network since fully-connected networks are computationally demanding [8 p. 366].

Softmax

Softmax layer is an activation function defined as

$$\varphi(z_i) = \frac{exp(z_i)}{\sum_j^K exp(z_j)} \tag{6}$$

for output z where z_i is the ith value from a vector of K elements. Softmax is typically used as the last layer of a network to represent a categorical probability distribution over a finite vector of K elements. In the case of semantic segmentation, softmax is computed for each output pixel and K is the number of output classes. Softmax is typically used in conjunction with cross-entropy to compute the loss function for segmantic segmentation, while argmax is used to get binary output labels.

2.4. Deep Learning and Image Processing

Deep learning is an umbrella subset of machine learning methods that leverage large, deep networks of artificial neurons. Deep neural networks have become particularly prevalent in pattern recognition tasks for their ability to represent arbitrary information such as context-dependent feature space mappings while also having a property of outputting information based on the information given to the network as an input.

Many computer vision problems have seen significant progress in the past decade due to the remarkable effectiveness of deep learning methods at solving pattern recognition tasks. In 2012, AlexNet, a deep convolutional neural network, ushered in a new era of machine learning where graphics processing units would be utilised to run gradient-descent based optimisation algorithms over a large number of parameters, with the number of optimisable parameters today being typically in the range of a million to a billion trainable parameters for a typical computer vision solution that is based on deep learning [19].

AlexNet achieved State-of-The-Art results on image classification in the ImageNet LSVRC-2010 [20] contest with a network size of 60 million parameters and 650,000 neurons. The network has five convolutional layers, three max-pooling layers, three fully-connected layers, and a softmax output layer. Figure 4 shows the structure of a typical image classification network. This structure is similar to AlexNet and has been proven to be particularly good at global image-level pattern recognition.

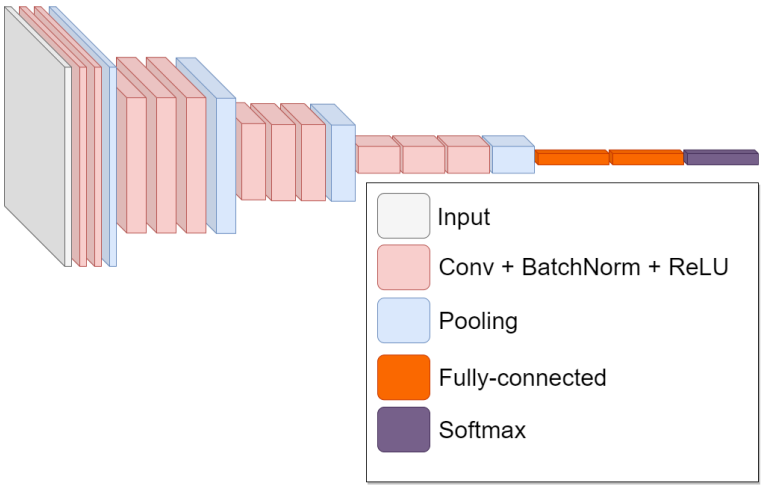


Figure 4. A deep neural network structure with a 2-D input such as an image and a 1-D vector output.

Since 2012, progress in deep learning-based computer vision solutions have been to a large degree by advances in computing power, deeper networks with larger amounts of parameters, availability of larger training sets, and network structures that have been designed to perform better at specific tasks. Krizhevsky et al. mention that their experiments suggest that their results in [19] can be improved simply by technological improvements in computational speed and larger datasets.

Empirical tests show that generally in deep learning more data translates to better results. Sun et al. show the tremendous capacity of deep learning methods to benefit from larger and larger amounts of training data, increasing the amount of data from ImageNet's 1 million images up to 300 million coarsely annotated images achieve significant improvements in classification accuracy [21]. While their motivation was to research feeding noisy data, this thesis is motivated by the potential increase in segmentation output accuracy by cost-efficiently generating more data with the help of machine learning.

2.5. Network Architectures

This section describes deep convolutional neural network archetypes that have been developed over the years to achieve different end-goals with focus on architectures that are relevant to neural networks that have been developed specifically for semantic segmentation. More specifically, encoder and autoencoder networks are discussed. A deeper look into semantic segmentation is provided in subsequent chapters.

Autoencoders

Autoencoder is a network structure that is traditionally used for feature extraction and dimension reduction by training the network to perform input reconstruction with an architecture and objective function that encourage learning encoding and decoding. One way to enforce learning dimension reduction is to constrain the dimensions of the encoder stage. Decoding of information is then learned by learning to effectively copy inputs to outputs while controlling that the network does not overfit to the training data. Figure 5 illustrates the autoencoder architecture. More modern works use autoencoder architecture for generative tasks such as denoising or semantic segmentation, although semantic segmentation architectures such as U-Net typically utilise additional structures such as learned upscaling and skip-connections in the decoding block to achieve higher accuracy [22]. Semantic segmentation autoencoders are described briefly in Chapter 3.

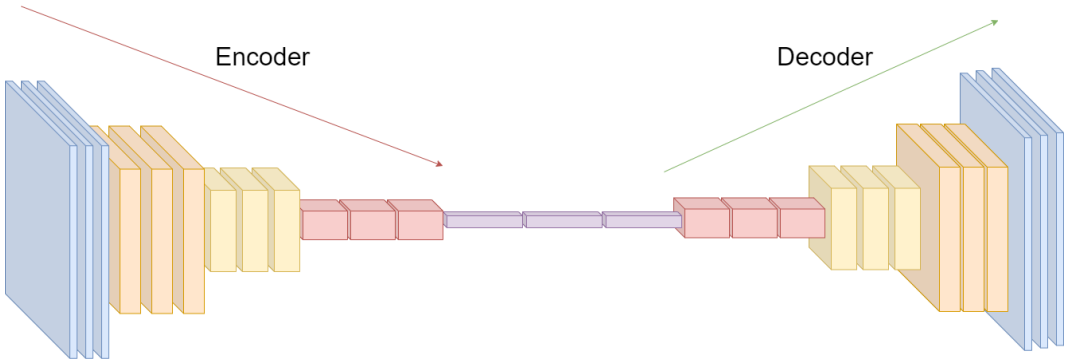


Figure 5. A fully-convolutional autoencoder neural network.

3. SEMANTIC SEGMENTATION

The goal of this chapter is to get an understanding of semantic segmentation and basic tools used to annotate semantic segmentation training data.

Semantic segmentation is a process where images are subdivided into areas such as similar regions or objects based on semantics present in an image by assigning class information into an image at pixel-level. It is described as an ill-posed task in that information from outside the image is required to correctly solve semantically uniform areas in all images. To illustrate this, suppose a person is wearing a jacket and a semantic decision that the jacket is a part of the person region, while the jacket is also semantically separate from the person. The semantic segmentation algorithm would then need to be able to determine that this jacket is a part of the person region. To make the decision more challenging, an image could also contain a jacket that is not worn by anyone, in which case it would not belong to a person region semantically.

This issue can be alleviated by introducing external information into the algorithm. In recent semantic segmentation methods, this outside information is introduced into the system as a collection of images in the form of a training data set. The world represented in digital images contains vast numbers of variations of this problem of semantic ambiguity, which in part makes it necessary to use sufficiently large training sets to reach useful level of semantic representation and accuracy for different use-cases with the current state-of-the-art methods.

Ambiguity of semantics and the requirement for external information about semantics make semantic segmentation traditionally one of the most challenging tasks in digital image processing. In applications that utilise semantic segmentation, success and failure are determined by segmentation accuracy. An autonomously driving car may not be able to react to a pedestrian if the pedestrian was not detected, an image stylization algorithm may not produce visually pleasing results if the image segmentation was not accurate, and a product line might not notice a faulty item if areas critical to the item are not correctly detected.

While some of the challenges with segmentation can be alleviated in controlled environments to produce robust results, segmentation out-in-the-wild remains challenging due to lack of control over the environment despite increasingly robust segmentation methods [23].

Modern semantic segmentation solutions that are based on deep learning are capable of semantically segmenting images into multiple predefined object categories and object instances within an object category. These solutions are limited by training data. Current segmentation neural networks are not capable of accurately recognising an object category that is not present in the dataset that was used to train the neural network. This thesis focuses on binary classification, classifying pixels as either part of a main subject person or as non-main subject image area.

3.1. Definition

As defined by Gonzales and Woods in [23] and extended for semantic segmentation, semantic segmentation of an image I can be defined as the union of all of its constituent regions R_i so that

$$I = \bigcup_{i=1}^{N} R_i \tag{7}$$

where N is the number of semantically meaningful areas in an image I such as "sky", "human" or "face" and R is a connected set defined so that

$$R_i \cap R_j = \emptyset, \text{ for all } i \neq j \tag{8}$$

$$Q(R_i) = TRUE, \text{ for all } i = 1, 2, ,,,, n \tag{9}$$

$$Q(R_i \cap R_j) = FALSE, \text{ for all } i \neq j \tag{10}$$

where Q is a logical predicate. The result of (9) Q(R_i) can be understood as the result of segmentation for region R_i where all points in R_i have the same value. Formula 10 indicates that different regions must not overlap.

The accuracy of a segmentation is typically measured in Jaccard index—also known as intersection-over-union, defined as

$$\frac{|y' \cap y|}{|y'| + |y| + |y' \cup y|} \tag{11}$$

where y' is the approximated output of the ground-truth y. This can be further interpreted as

$$\frac{TP}{TP + FP + FN} \tag{12}$$

where TP, FP and FN are True Positive, False Positive, and False Negative respectively.

This work focuses on binary portrait semantic segmentation demonstrated by Figure 6. That is, the semantic segmentation takes into account two image region categories, namely the main subject region of an image and non-main subject regions. More specifically, in this thesis, an image I is defined as

$$I = F \cup B \tag{13}$$

where F is the main subject region or foreground and B is the non-main subject region or background.

This work also uses alpha matting as a part of the implemented process, which defines an image I as

$$I_i = \alpha F + (1 - \alpha)B \tag{14}$$

where α is a value in range [0.0, 1.0] and indicates the transparency of a foreground region pixel, so that opaque areas such as strands of hair will get some value between 0 and 1. Alpha matting can further be defined as

$$I = \sum_{i}^{K} \alpha_i F_i, \text{ where } \sum_{i}^{K} \alpha_i = 1 \tag{15}$$

for K components.

3.2. Semantic Segmentation as Annotations

This section describes the conventional process of creating semantic segmentation annotations.

A typical solution to the problem of semantic ambiguity is to use annotations created by humans as a target output of the segmentation algorithm. These annotations are usually stored either as a dense pixel representation in the form of an image file (Figure 3.6(b)) or as a sparse polygonal representation (Figure 7). Storing the annotation as an image file has the benefit of being compatible with all image viewers and image processing tools for ease of access. Polygonal representation is by nature sparse, which is efficient when storing the annotations, but special tools are required to decode the polygons for visual inspection and manipulation. Annotation tools exist for both representation types. Some of the most notable segmentation annotation tools are described briefly in Chapter 4.

Semantic segmentation is typically annotated by manually drawing outlines around objects with the chosen annotation tool. These include polygons, free-draw, and computationally assisted methods such as superpixels [24]. Newer annotation tools also support active learning where the output of a deep neural network is used as the initial annotation which can then be corrected manually by an annotator [25, 26].

Figure 6 illustrates semantic segmentation and its' one of its' use-cases. While Figure 3.6(b) shows the output of a semantic segmentation system, the figure could also be a ground-truth annotation, as ideally the output of a semantic segmentation system would be identical to its ground-truth annotation. This raises the main question in this thesis, can the output of a semantic segmentation system be used to efficiently reduce the amount of manual labour required to create ground-truth annotations in large scale when the goal is to improve the performance of the same segmentation system?

Fully manual annotation

Most common annotation tools for fully manual annotation are polygonal and free-draw drawing tools. The main difference between these is that polygons are annotated sparsely as points around the object, so an annotator only needs to click a relatively low number of times on the object instance border to fully annotate the object and increasing the number of points typically increases the accuracy of the annotation for complex shapes. With free-draw tools an annotator needs to draw the edge line pixel-by-pixel around the object instance which requires the cursor movement to be consistently accurate along the edge line. Figure 7[1] visualises a polygonal segmentation annotation. It has been shown that polygon annotation is roughly 1.5 times faster than freedraw while maintaining roughly the same accuracy [24, 2].

[1] Photograph by Otso Suvilehto

(a) Image 1 (Photograph by Mídia Ninja at Flickr https://www.flickr.com/photos/midianinja/16523696796/ Licensed under CC-BY-SA 2.0).

(b) Segmentation of Image 1

(c) Image 2 (by Pixabay at Pexels https://www.pexels.com/photo/big-ben-bridge-castle-city-460672/ Licensed under CC0 1.0).

(d) Image 2 + Image 1 masked with Segmentation

Figure 6. Image stylization with semantic segmentation.

Figure 7. A polygonal semantic segmentation annotation.

Computationally-assisted annotation

Computationally-assisted annotation tools consist of a wide range of tools that are based on different algorithms such as superpixels, alpha matting or CNN outputs. Figure 8 visualises a graph-cut-assisted annotation with corresponding superpixel regions calculated with graph-cut in Matlab Segmenter toolbox. Red lines represent regions that have been manually labelled as background and green lines represent areas manually labelled as foreground. It has been shown that annotation based on superpixels is roughly 2.8 times faster than free-draw while maintaining roughly the same accuracy [24].

Another topic of assisted annotation is active learning, where an algorithm is used to estimate which samples should be annotated for optimal annotation efficiency [27]. This thesis treats active learning as a by-product of the manual step in annotation with automatically generated annotations categorised as good, fairly good, or bad instead of relying on an algorithm to select samples. A small portion of annotations categorised as bad is passed for manual correction in every second iteration. The effects of selecting a small number of images for manual annotation based on the segmentation algorithm is discussed in subsequent chapters.

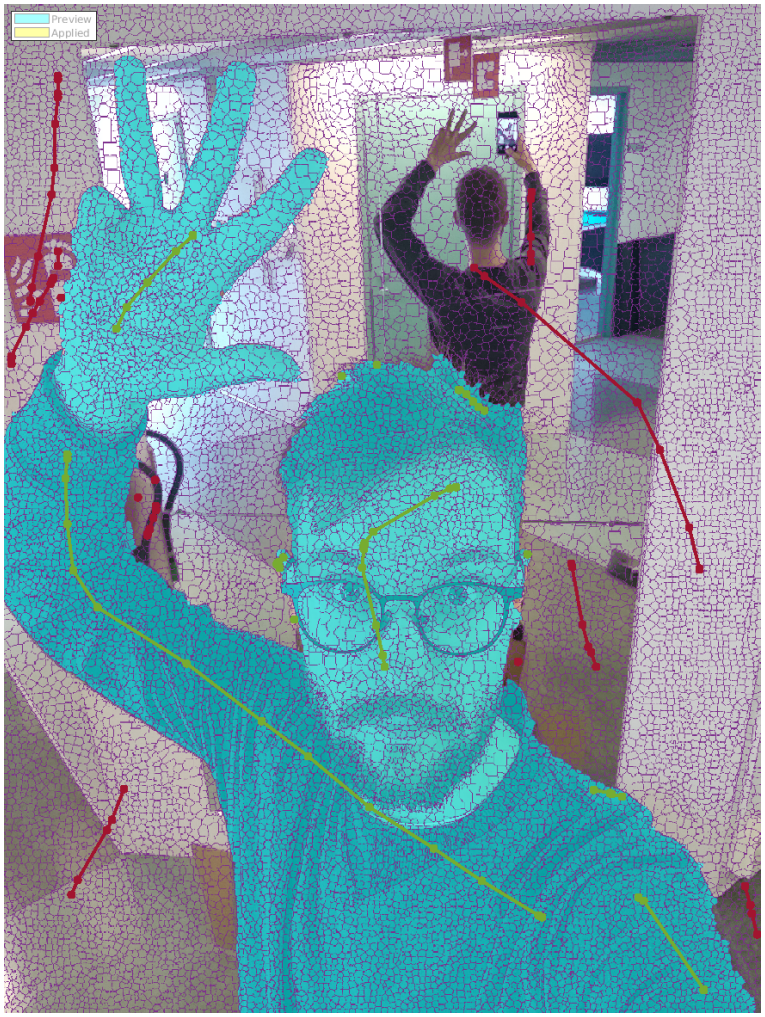


Figure 8. A semantic segmentation annotation assisted by graph-cut.

3.3. Semantic Segmentation with Neural Networks

Convolutional neural networks have gained popularity in recent years in image processing tasks for their ability to generalise well to traditionally challenging image analysis tasks, often performing significantly better than traditional methods and reaching State-of-the-Art results in these tasks. This also became true for semantic segmentation in 2015 when Long et al. published their results in [16], achieving State-of-the-Art results in Pascal Visual Objects in Context challenge segmentation task by a significant margin. Figure 9 visualises the general network architecture used by Long et al. Convolutional neural networks have a history at least back to 1991 where Matan et al. used a convolutional neural network with four convolutional operations in multi-digit recognition [28].

Neural networks are particularly interesting for semantic segmentation for their ability to learn arbitrary representations from complex sets of data. Recent machine learning semantic segmentation solutions rely heavily on deep fully convolutional neural networks. Convolutional neural networks are networks that use at least one convolution or a cross-correlation operation in place of standard matrix multiplication

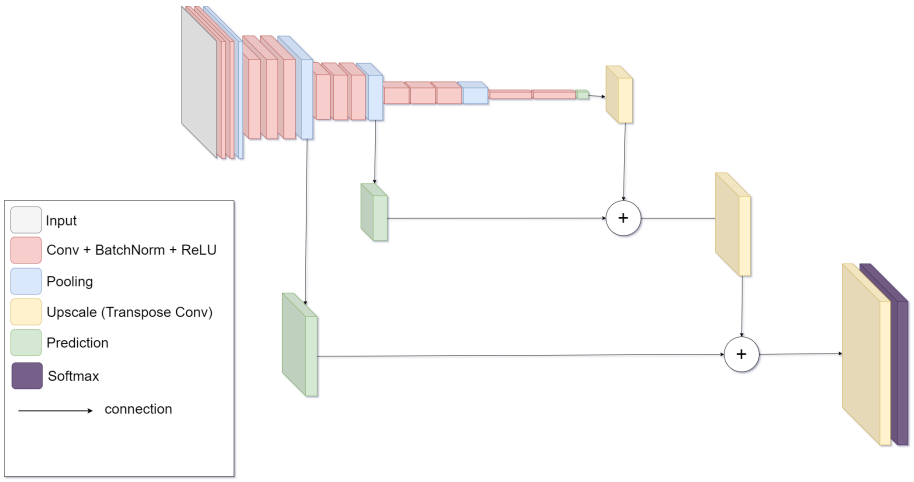


Figure 9. Fully convolutional autoencoder neural network FCN8s.

and typically share characteristics with autoencoder network structure visualised by Figure 5.

In autoencoders, the encoder half decreases the resolution while introducing layers with increasing number of convolutional weight kernels thus increasing the width of the layer. The decoder half typically works in the opposite way by performing upscaling and using fewer convolutional weight kernels to balance computational cost. Fully Convolutional Neural network FCN8s can be compared to autoencoders with the difference that FCN8s performs predictions (green in Figure 9) and performs learned upscaling of the class outputs with transpose convolution in three steps, summing the results along the way. Figure 10 shows the general architecture of U-Net, a popular autoencoder for semantic segmentation [22]. U-Net is an autoencoder type network architecture introduced by Ronneberger in [22] for semantic segmentation of biomedical images in 2015. It introduces skip-connections to the autoencoder architecture for improved resolving and propagation of higher-level features in the decoder block. Many semantic segmentation networks utilise ideas similar to FCN8s and U-Net where information from the encoder is passed to the decoder via skip-connections to preserve higher-level information found at higher resolutions in the encoder.

Deep semantic segmentation neural networks are commonly unified by a large number of trainable parameters, which in turn translates to a large capacity to learn and to risk of over-fitting to a small training set. This characteristic makes pursuing larger training datasets appealing if computational capacity is not an issue.

This work uses a semantic segmentation method called DeeplabV3+[2] introduced by Chen et al. in [29]. DeeplabV3+ is a convolutional neural network that introduces an atrous spatial pyramid pooling module (ASPP) in the decoder stage of the network which can be used with different encoder network structures.

[2] https://github.com/tensorflow/models/tree/master/research/deeplab

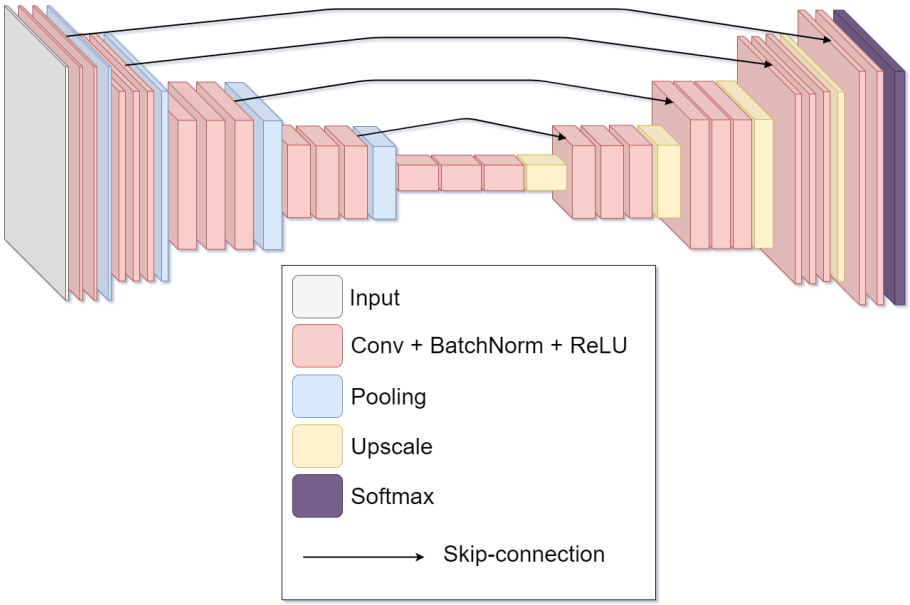


Figure 10. U-Net: A fully-convolutional autoencoder network with skip connections.

4. SEMANTIC SEGMENTATION DATASETS

This chapter takes a deeper look into the process of creating a semantic segmentation dataset and briefly describes some notable semantic segmentation datasets and the tools and amount of work required to create them.

4.1. Data Acquisition

Deep learning and its' need for large quantities of training data have opened up a market for curated data collection services to fill the needs of commercial deep learning applications with the benefit of not having to allocate large amounts of work hours into curating training data. These Training-Data-as-a-Service (TDaaS) include services to take tightly specified types of images captured either by the service provider's in-house photographers, crowd-sourced image collection platforms where users of the platform upload images that they have taken and receive a monetary compensation for each image they upload, as well as services that collect data on the internet from online platforms such as Flickr[3] and Getty Images[4] that publish images under the Creative Commons license. The service fee is relative to the amount of manual labour required to curate the data to customer's specifications and can get prohibitively costly for large orders.

An alternative to these services, often preferred by the scientific communities, is to use websites such as Flickr and Getty Images to search and download royalty-free photographs posted under the Creative Commons license with specific keywords to construct a dataset. Some of the most notable publicly available semantic segmentation datasets utilize Flickr which was popularised by Pascal Visual Object Challenge as their source of image data. Using online platforms as a data source has the benefit of getting access to images taken by a large number of photographers everywhere around the world, although copyright-licenses must be taken into account in the usage and redistribution of data. However this does not automatically guarantee an unbiased dataset, as bias can be introduced with search keyword selections, and photographers tend to take similar pictures of similar subjects [30, 31]. Everingham et al. suggest that data selection bias can be alleviated to some extent with context-related keyword selection, searching for contexts that typically include instances of desired things. For example, searching for "town square" would quite likely yield images that contain people, stalls and vehicles and would provide varying point-of-views to said categories [31].

4.2. Quality Control

Quality control is an important part of any process, and the quality control of deep learning datasets is a large field of research. This section acknowledges the effort required to control the quality of data when creating a semantic segmentation dataset. The annotation process in this thesis does not address quality control, and as such

[3] `https://www.flickr.com`

[4] `https://www.gettyimages.com`

the description is kept brief with the focus being purely on controlling the quality of annotations, though arguably some of the discussion on quality control could also be applied to data collection.

Many different kinds of processes, tools, and ways of employing annotators are used in the creation of semantic segmentation datasets. As such many different quality control processes also exist to ensure sufficiently high level of annotation accuracy on image-level and across the dataset depending on criteria such as accuracy requirements, budget, or available tools and workforce. This section discusses quality control in terms of the amount and type of available workforce. More specifically, here quality control is split into processes performed by experts and by the crowd.

Quality control implemented as an expert task typically trusts that the experts performing the quality checks can be trusted. For example in the case of Pascal Visual Objects Challenge (VOC) 2007 all annotations were checked by one of the annotators [32] as described in section 4.4.1. In later revisions of Pascal VOC quality control was performed in parallel with segmentation annotation, and examples of common segmentation mistakes were shown to workers performing quality checks. Over time, it was reported that quality checks took around 50% of the workers' time [33].

Quality control can also be performed by the crowd such as in the case of Microsoft Commmon Objects in Context dataset [24] by Lin et al. described in section 4.4.2. Lin et al. take into account in each step of the annotation process that crowd-sourced annotations may have large variance in accuracy between annotators. To minimise the risk of low-quality annotations, multiple people were involved in the annotation of each image. Instance spotting was performed by eight workers on each image, and in instance segmentation each instance was annotated by only one worker to minimise costs. Lin et al. reported that a union of eight workers at Amazon Mechanical Turks achieved a higher recall than any of their expert workers in the evaluation stage [24].

To reduce segmentation errors, each worker was required to complete a training task on each object category that they were to annotate. Each instance segmentation was then verified by three to five workers to check that the instance outlines are drawn accurately. Lin et al. report that only one in three workers passed the training stage, but some annotators remained to consistently submit inaccurate annotations. All annotations by such workers were discarded [24].

Overall, many variances in quality control processes exist, and the process that is selected depends on the available workforce and budget. If the trust factor on workers is low, the quality control process may need to be more rigorous to ensure acceptable annotation accuracy.

4.3. Common Labeling Tools

Tools used to label semantic segmentation annotations include a wide range of fully manual and computationally-assisted annotation tools and workflows. Some publicly available semantic segmentation datasets have been released alongside the tool that was used to create the dataset while others use commercial graphics designing tools such as Adobe Photoshop, or customise their own tools from open-source annotation tools to fit their needs and annotation workflow.

This section describes some tools from each category with focus on tools that were used to create datasets described in Chapter 4.

4.3.1. LabelMe

LabelMe is web-based annotation tool and database initially published by Russell et al. in 2008 [34]. LabelMe provides a polygon drawing tool, a way to assign labels to objects freely, and an image query database for an easy way for annotators to move on to the next image. LabelMe leaves annotation quality control to the user. LabelMe stores annotations as XML which makes them easy to store and modify. Figure 11 displays the user interface of LabelMe which provides basic polygon, eraser, and rectangle tools for annotation. Additionally, LabelMe provides a tool for scribbling mask areas, although this tool outputs a rectangular annotation instead of a mask polygon despite intermediate results showing a mask (Figure 12, 13).

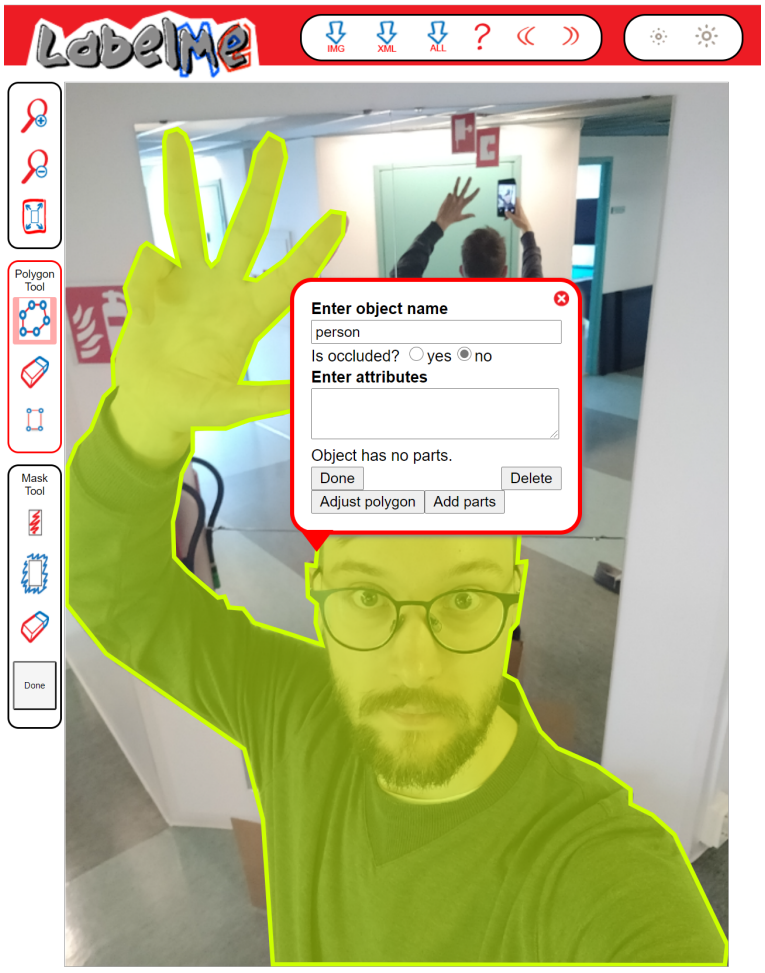


Figure 11. A polygonal semantic segmentation annotation done in LabelMe.

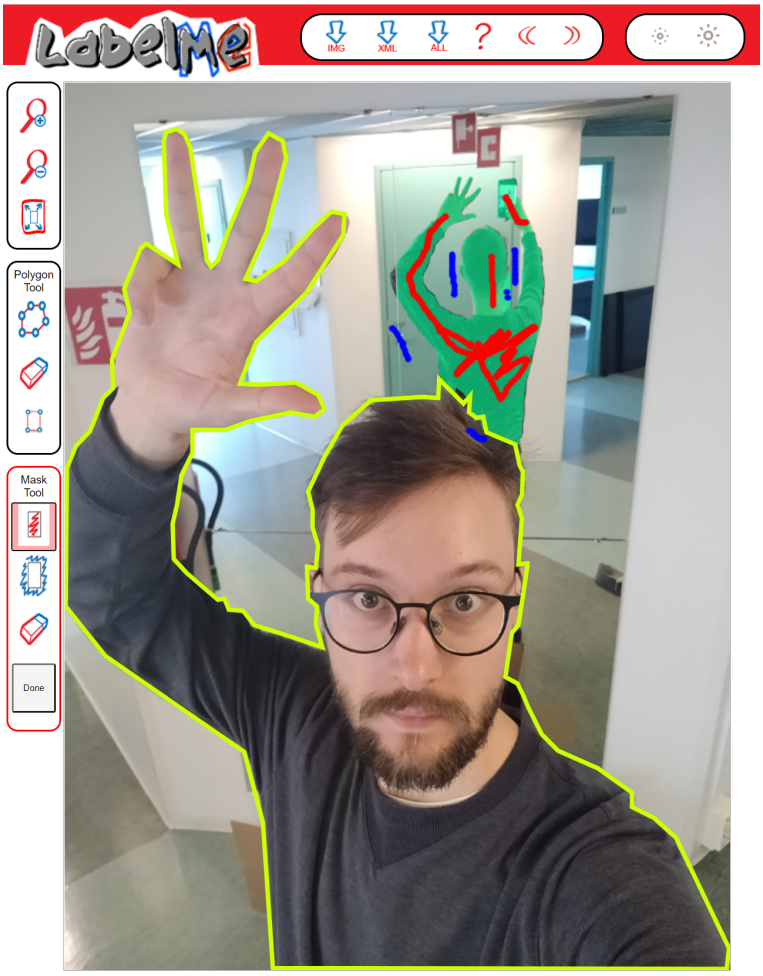


Figure 12. The mask tool in LabelMe visualises results as a segmentation mask.

4.3.2. Adobe Photoshop

Adobe Photoshop is a powerful image processing software widely used by photographers, graphics designers, and artists. While it is not specifically designed for annotation work, it has been used to create semantic segmentation annotations in datasets such as PortraitFCN+ for its' powerful interactive segmentation tool. Particularly Photoshop's Quick Selection tool has been mentioned in datasets such as [35, 36, 37] for its ease of use.

4.3.3. Matlab Labelling Tool

Matlab provides a semi-automatic segmentation labelling tool where label areas are marked, and the tool extends these areas to the nearest edges. It works reasonably well when object boundaries are very distinct from the rest of the image, but is prone to segmentation errors at object boundaries when the difference between the pixels in the object and outside the object is not sufficiently large. Manually correcting inaccurate edges may also be difficult with Matlab Segmentation labelling tool. Figure 8. shows annotation results using Matlab's graph-cut segmentation tool. Figure 14, 15 and 16

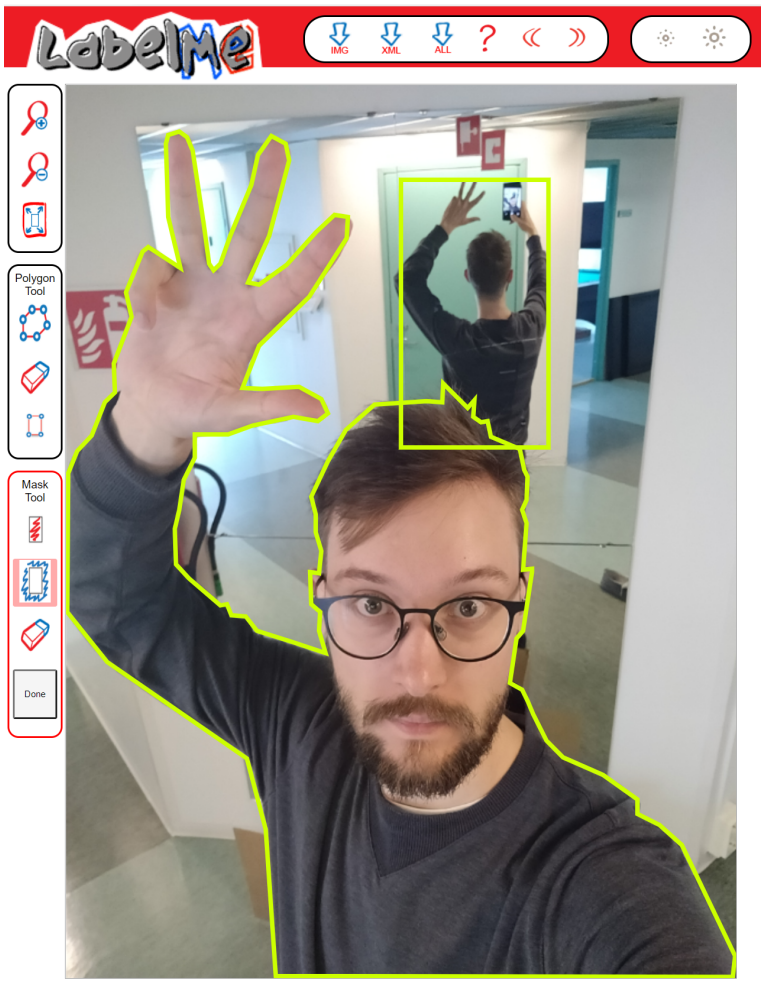


Figure 13. Masking results in a rectangle.

shows a particular behaviour with Matlab's graph-cut segmentation tool. Figure 14 shows the initial annotation in which should be noted that the arm in the background is included in the annotation. Figure 15 shows that excluding the arm in the background also excludes a portion of the annotated foreground even though the areas are far-apart spatially. Figure 16 shows that this can be corrected with an additional scribble in the foreground. This was one of the contributing factors to why a fully manual annotation tool was selected for manual correction in this work.

kuvia

4.3.4. Microsoft Visual Object Tagging Tool

Visual Object Tagging Tool (VoTT) [25] is an open-source annotation tool for image and video media. It was originally developed by Microsoft Commercial Engineering Group in Israel. VoTT provides tools for drawing rectangles and polygons and assigning categories to each annotation. It also provides various export format options such as Azure Custom Vision Service, Microsoft Cognitive Toolkit, Tensorflow (Pascal VOC and TFRecords), generic JSON schema, and Comma Separated Values. VoTT supports Azure cloud, Bing image search, and local file systems for importing data.

Figure 14. Initial scribble.

VoTT is available on Linux, Windows, OSX, and it is available as a Web application on modern Web browsers [38]. Figure 17 shows the user interface of VoTT.

4.3.5. OpenCV Computer Vision Annotation Tool

OpenCV Computer Vision Annotation Tool (CVAT)[26] is an open-source annotation tool for image and video media. It was released in 2019 by Intel as a part of OpenCV open-source image processing library to speed up the process of annotating image and video media for use in training machine learning applications. It provides tools for labelling images with rectangles, polygons, polylines, and points as well as assigning categories to each annotation. It also provides features for automating typical annotation tasks. These automation tools include an option to use the Tensorflow object detection API to automate object annotation as well as interpolation of annotations between keyframes in a video feed. It also has a semi-automatic segmentation mode where an object instance boundaries are annotated with a few points and an algorithm then resolves the instance segmentation. CVAT has a Web application interface and it supports sharing work between teams [39].

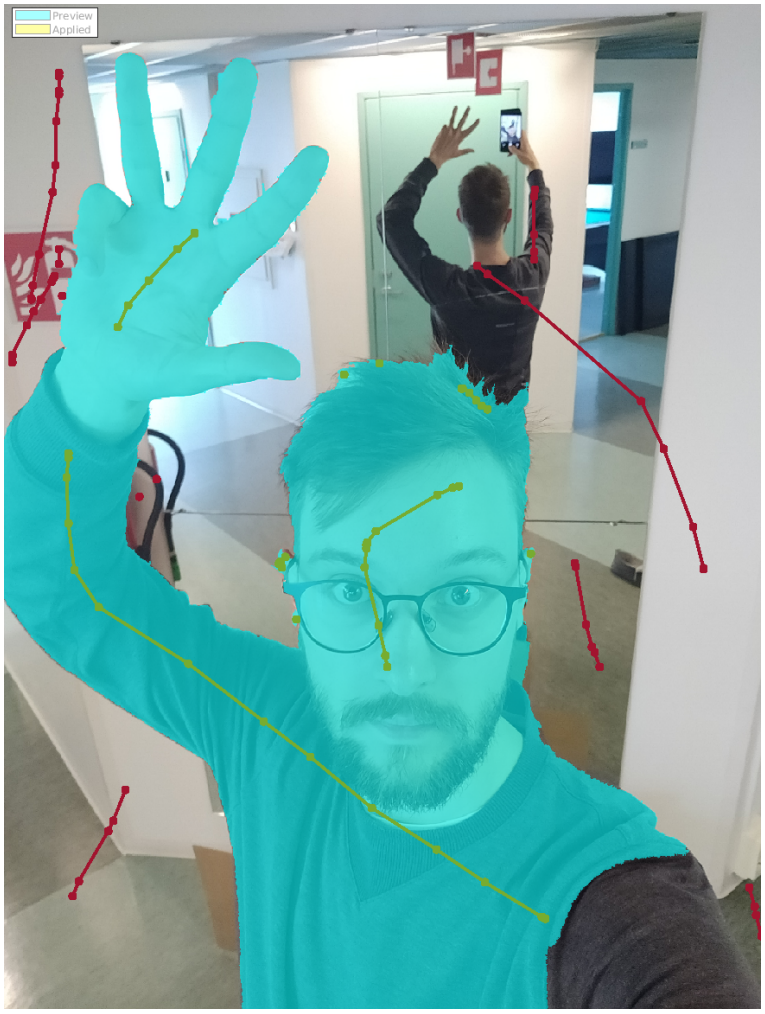


Figure 15. Excluded arm in the background.

4.4. Common Semantic Segmentation Datasets

This section aims to give a comprehensive understanding of the work that goes into creating large semantic segmentation datasets and explain the motivation for larger and more complex datasets.

Some of the most common semantic segmentation datasets are described briefly.

4.4.1. Pascal VOC

Pascal Visual Object Classes is a visual object dataset that contains multiple types of annotations and has received multiple improvements and increases in size over the years. At present, the latest and final version of Pascal VOC (2012) contains 20 classes and 11,500 images in the training and validation set with 27,450 object ROI annotations and 6,929 semantic segmentations. The listed classes are exclusively "things" such as airplane, train, sheep, potted plant, and person. Context categories such as sky, forest, and road are not included in the Pascal VOC dataset, although a separate dataset named Pascal-Context has been created for detection and segmentation

Figure 16. Included arm in the foreground.

of such categories and contains an additional 520 categories for semantic segmentation and object detection.

Original Visual Object Classes 2005 [40] challenge dataset has 4 classes, which was increased to 10 in VOC2006 [41], and to 20 in VOC2007 [32]. Images for the VOC challenges were downloaded from Flickr with an automated program with listed keywords for each class. Image selection and keyword selections were performed at random.

Images for Pascal Visual Object Classes 2007 challenge were annotated in an annotation event where all annotators were given guidelines and provided training and occasional supervision. All annotations were then checked by one of the annotators to ensure high accuracy and completeness. The 2007 dataset was reported in the 2009 paper to have received only one error report [31].

To save annotation time, a 5-pixels wide "void" boundary was allowed between the object boundary and the segmentation boundary. The 2007 challenge dataset has 422 images with semantic segmentation annotations and 1,215 segmented object instances. Additionally, the test set contains 210 images with 607 object instances.

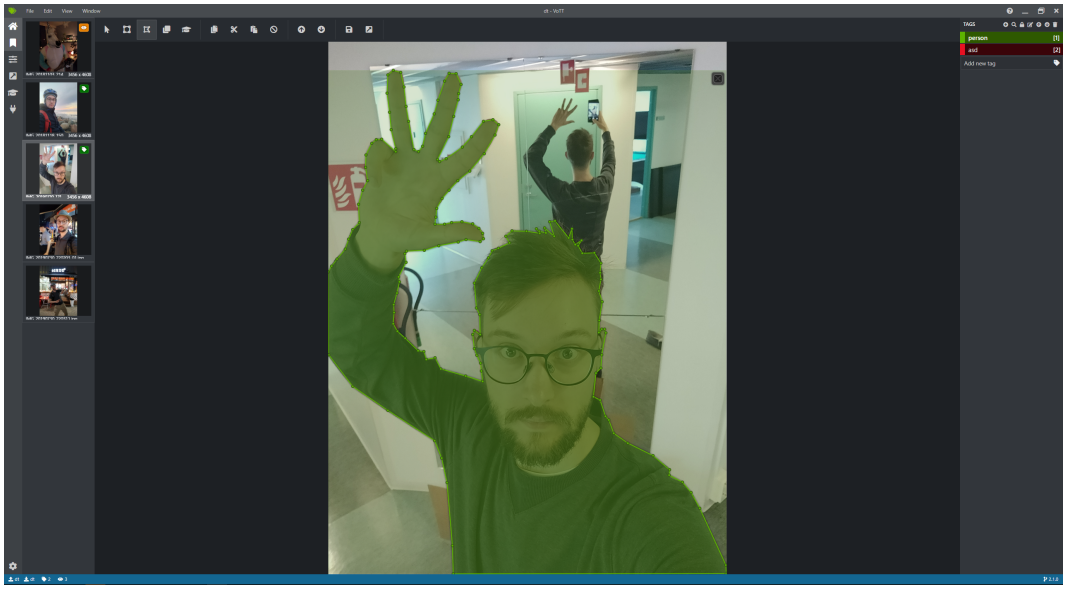


Figure 17. A polygonal semantic segmentation annotation done in VoTT.

The number of images with segmentation annotations has since then been increased to 6929 in 20 object categories in VOC2012 train/val set, and including the private test set, to 9993 images in total.

Mottaghi et al. labelled every pixel in the VOC2012 set to a category, providing 520 additional categories for semantic segmentation and object detection. This set is commonly named Pascal-Context [42].

These additional annotations are reported to have been done by six in-house annotators over the course of three months. They reported achieving significantly higher accuracy with in-house annotators than with crowdsourcing. Each image had on average 12 regions, and each annotator spent from 3 to 5 minutes on an image. Their tool provided an interface similar to LabelMe. The user-interface of LabelMe is described later in Chapter 4.

4.4.2. Microsoft COCO: Common Objects in Context

Microsoft COCO: Common Objects in Context is an important semantic segmentation dataset with 328,000 annotated images and more than 2.5 million annotated object instances in 91 object categories. 82 of the categories were reported to contain more than 5000 object instances in the dataset with the least instances in categories "hair drier" and "hairbrush". The categories were constructed to also be fully backward-compatible with Pascal VOC, so all 20 categories in Pascal VOC are included in COCO. The authors of Microsoft COCO group images into three categories, iconic-object images, iconic-scene images, and non-iconic images [24]. Iconic images are described as images of a specific category such as "a photograph of a toothbrush". Non-iconic photographs are described as images that contain instances of a category such as "a photograph that includes a toothbrush", in which the scene context could be something completely irrelevant to said category, for example a photograph of a festival with a toothbrush visible somewhere in the scene. In short, in iconic images,

the focus is on the category and in non-iconic photographs, the focus is on something else. Microsoft COCO dataset focuses on non-iconic images, which helps with better generalization to a task [30]. The images were collected from Flickr and downloads were limited to 5 images from the same uploader within a short period of time.

Microsoft COCO's image annotation process contains multiple stages and multiple annotators per image. The first stage is category labelling, where images are labelled with image-level categories of the objects that the image contains. The second stage is instance spotting, where each labelled instance is spotted and marked on the image. The third stage is segmenting the marked instances, and finally checking for instance segmentation quality. Annotations are created using crowdsourcing with Amazon Mechanical Turks. Object labels are drawn with polygons. Stuff categories utilise object labels and superpixels [43] for speeding up manual work. Caesar et al. report that superpixels are 2.8 times faster than their free-draw reference while being about as accurate, as the accuracy difference between free-draw, polygon and superpixels is similar to annotation accuracy within a single annotation type, even when repeated by a single annotator.

Category Labeling Stage

Microsoft COCO authors group the 91 categories into 11 super-categories [24]. Annotation workers would then label the images with information on whether or not each image contains instances that belong to any of the super-categories. Each image was labeled by eight workers to ensure high recall. False-positive labels are handled in later stages.

The category labelling stage was reported to have taken approximately 20,000 work hours to complete for the total of 328,000 images, which translates to roughly 16.4 images per work hour.

Instance Spotting Stage

The first stage provided locations for instances, but multiple object instances of each category may be present in an image. Each image was labelled by 8 workers, and each worker was asked to label at most 10 instances of each object category. This stage was reported to have taken approximately 10,000 work hours to complete which translates to 250 images per work hour.

Instance Segmentation Stage

Workers were presented with instances marked in previous stages and were asked to draw segmentation labels for those instances. Over 22 work hours per 1,000 instances was reported in this stage. This would translate to a total of 55,000 hours and 45 images per hour. This stage presented problems with annotation quality which would need to be taken care of with worker training and additional quality control procedures.

4.4.3. CityScapes

CityScapes is a large semantic segmentation dataset that consists of stereo video sequences recorded on the streets in 50 different cities. The dataset was published by Cordts et al. in 2016 and it is aimed towards self-driving cards [2]. 5000 images have been annotated with high-quality pixel-level annotations, and additional 20,000 images have coarse segmentation annotations. In coarse labels, annotation edge accuracy was traded for annotation speed [2].

Objects in CityScapes images were annotated back-to-front, so that objects farthest from the camera were labelled first. This was reported to ensure a consistent boundary between occluded and occluding object instances. Polygons were labeled by in-house annotators using LabelMe annotation tool, and high quality pixel-level annotations were reported to have taken 1.5 hours per image. Coarse annotations were reported to have taken on average 7 minutes per image. The only constraint in the coarse annotations was that each pixel within a polygon must belong to the same object category.

Cordts et al. reported a quality control test, where 30 images were annotated twice by different annotators. The pixel-level annotations were found to have a 96% overlap, and if uncertain "void" regions were ignored, the overlap between different annotators was found to be 98%. Additionally, 97% of pixels in the coarse annotations were found to have been labelled into the same category as in corresponding high-quality annotations of the same image [2]. These accuracy results by Cordts et al. are similar to what was reported in [43] and will be noted in the evaluation section of this research.

4.4.4. PortraitFCN+

PortraitFCN+ is a portrait segmentation dataset released by Shen et al. alongside their work "Automatic portrait segmentation for image stylization" [35]. The dataset contains around 1800 images collected from Flickr and is available for academic use on request. PortraitFCN+ dataset has been widely used as a benchmark on portrait segmentation research [44, 45]. The PortraitFCN+ dataset is used as the starting point in this research, which makes it the most relevant dataset described in this chapter.

PortraitFCN+ dataset is strictly an upper-body portrait image dataset. Image orientations are set to portrait with 600-by-800 pixels, and the downloaded images are cropped to upper-body based on the size of face detection bounding boxes. In addition to cropped portrait images, the dataset includes three additional channels, namely an affine warped mean mask of the total dataset and centre of the face position on x and y channels, with the centre of the face being normalised to value 0 on x and y axes respectively. Shen et al. report that introducing the additional channels improve segmentation accuracy, but the channels are not used in this research despite a promise of greater performance.

The data was annotated by students using Photoshop Quick Selection tool. Shen et al. provide no additional statistics on the dataset creation.

4.4.5. Supervisely Person

Supervisely Person is a person segmentation dataset, in which all people in an image are annotated. The dataset was released in a blog post in March 2018 by Supervise.ly[5], and is publicly available and free for academic use [46]. Their work serves as motivation for this research and shares this research utilises an iterative workflow similar to the one used by Supervise.ly with 2 additional steps for finer annotation edge accuracy.

Supervisely Person dataset was done by 2 workers, and the whole process took 4 days. The tool used was Supervise.ly segmentation tool(2018 version). The resulting dataset contains 5711 annotated images with 6884 annotated object instances [46]. Their annotation workflow consists of a neural network for initial annotations, manual validation, and correction.

Supervise.ly merged Pascal VOC and Mapillary datasets and selected only images that contain humans for an initial training set. Their annotation workflow consists of a neural network for initial annotations, manual validation, and correction. They downloaded around 15,000 images with tags related to person segmentation for the annotation process.

The validation section assigns initial annotations to three categories: bad annotation, annotations to correct, and good annotation. Images that have been assigned for manual correction are corrected, and corrected annotations and good annotations are added into the initial training data for a new iteration. This process is repeated until enough images have been annotated. Supervise.ly reported that in the first iteration, only 20% of the initial annotations were categorised either for correction or as good. After three iterations, this number increased to 70%. They performed 6 iterations in total. The resulting dataset contains 5711 annotated images with 6884 annotated object instances.

The workflow used by Supervise.ly is similar to the workflow introduced in this thesis, with the main difference being that this thesis uses additional computational steps and performs fewer manual corrections. Supervise.ly's work serves as a major motivation for this research on evaluating semi-automatic segmentation annotation processes.

4.4.6. Synthetic Depth-Of-Field with a Single-Camera Mobile Phone

This section goes over the annotation generation and data collection processes described in "Synthetic Depth-of-Field with a Single-Camera Mobile Phone" [44] by Wadhwa et al. briefly.

Wadhwa et al. mention using a semi-automatic segmentation annotation generation process and report increased accuracy over PortraitFCN+ dataset using their generated data. Their accuracy reports provide motivation for this thesis.

For their dataset, around 122,000 images containing from one to five faces were downloaded from Flickr and polygons were drawn around people in the images. The

[5] `https://www.supervise.ly`

polygons were then refined with Fast Bilateral Solver by Barron and Poole [47]. This thesis also uses Fast Bilateral Solver to refine the annotation boundary.

Wadhwa et al. describe augmenting their dataset with an additional 30,974 portrait images augmented with 13,337 background images at random. This additional data was downloaded from Flickr. Wadhwa et al. compute an alpha matte using KNN matting by Chen et al. [48] and composite the image onto 15 new backgrounds. While Wadhwa et al. use KNN matting to improve image augmentation, the workflow in this thesis uses KNN matting to create initial annotations from trimaps produced by a segmentation network. Wadhwa et al. report that improvements to the dataset were made over a course of nine months.

Wadhwa et al. report their neural network's prediction accuracy on the PortraitFCN+ evaluation set to be 97.01% when trained on PortraitFCN+ data, and 97.7% on the PotraitFCN+ evaluation set using their own training data [44]. The data created and used by Wadhwa et al. is not publicly available.

5. IMPLEMENTATION

The implementation in this work is an iterative semi-automatic semantic segmentation workflow with focus on reducing the amount of manual labour in comparison to fully manual annotation processes while ensuring that the annotations are of sufficiently high quality.

The main features of the workflow are semantic segmentation using a convolutional neural network that has been trained to the task on an initial dataset, refining the network outputs using an alpha-matting algorithm, and manually inspecting the generated results to decide whether they can be added to the training set or not. This chapter describes the implementation and each step of the workflow in detail.

5.1. Data Acquisition

The data used in this work was collected from Flickr filtering for Creative Commons and commercial usage with selected keywords that would likely yield images of people in various environments and poses. The selected keywords were "person", "people", "festival", "pose", "costume", "crowd", "ethnic", "fashion", "festival", "friend", "glamour", "glasses", "hands", "handsign", "hat", "man", "woman", "model", "modelling", "portrait", "salute" and "selfie". As this work is part of a larger research project the total number of images downloaded is roughly 100,000 for later use. The images were then cleaned using face detection with a minimum requirement that the largest detected face ROI must have width or height that is at least 12% of the larger dimension of the image. Additionally images were filtered by resolution so that the image size s is $max(width, height) >= 512$.

An additional dataset of drawings, paintings, and statues were downloaded from Flickr to create a dataset for detecting whether the face is a photograph of a person or some other media. Going into detail on the face media type selection is outside the scope of this work. This dataset was used to clean up the downloaded set of images.

5.2. Initial Data Selection and Ground-Truth Creation

After detecting images with sufficiently large faces, the faces were classified into two media types using a classifier network to prune out images of paintings, drawings and sculptures. The network classifies faces into two classes, ”photograph of a person” and ”other”. ”Other” includes media such as paintings, drawings, and statues. After pruning the images based on resolution, face size and face media type, the remaining set of images is roughly 27,000 images. 2010 images were selected at random from this set for annotation. The remaining images are later annotated using the same process, although without comparisons to manually annotated ground-truths.

The ground-truth annotations were created by a company that provides image annotation as a service. The annotations were drawn completely manually using Adobe Photoshop, following annotation specifications provided to them. These specifications are described in the next section. The company charged $0.8 USD per annotated person in an image, so that for example if an image contains three people but only two of them

are considered to be the main subjects, the annotation of that image cost $1.6 USD. The annotations were delivered at roughly 500 images per week.

Additionally PortraitFCN+ dataset collected and annotated by Shen et. al in [35] was used as the starting point which is then extended semi-automatically to contain a wider range of different scenes, lighting conditions, and compositions.

5.3. Annotation Specifications

The annotation task in this work is specified as a particularly ill-posed segmentation task where the ground-truths are highly subjective by design. The task is to annotate the main subjects and any items they may be holding or using in images that contain an arbitrary number of main subjects and non-main subjects. Annotators were given tasks to choose which people are the main subjects of an image in their opinion and to annotate them in each image. Visual inspection of each annotation was agreed on as quality control in the ground-truth annotation process to avoid the unreliable nature of quality control based on random sampling which was offered by the TDaaS service provider. More insight into the annotation service provider's internal processes is not available.

5.4. Semi-Automatic Annotation Generation

The software implementation of the annotation generation workflow can be divided into three parts: Semantic segmentation, alpha-matting, guided filtering, and thresholding. These three parts can be done fully automatically, and the last step of annotation generation is manually categorising the candidate annotations into good, fair, and poor based on accuracy via visual inspection. Parameters for each algorithm were acquired via optimisation as described later in this section.

5.4.1. Semantic Segmentation

In this context, semantic segmentation is used to generate trimaps for use in alpha-matting. The architecture used is DeeplabV3+ with xception65 feature extractor, and the network is always trained before each annotation iteration. DeeplabV3+ with the xception65 backend was chosen for this research because of its' high performance in the task of semantic segmentation and availability for use in research.

The segmentation network is always trained for 120,000 steps with publicly available weights [6] trained on Microsoft COCO and Pascal VOC as the initial weights. The batch size is set to 8 at 512x512 resolution and the network is trained on 2 GPUs so that the minibatch size on each GPU is 4. In this work, the training batch size is limited by available GPU memory. DeeplabV3+ output stride is set to 16 in training and in inference. Initial learning rate is set to 1e-2 with a schedule set to decrease by an order of magnitude every 40,000 steps.

[6] `http://download.tensorflow.org/models/deeplabv3_pascal_trainval_2018_01_04.tar.gz`

Once training is complete, softmax outputs of the network are then thresholded into trimaps for use in KNN matting. These thresholds are optimised with random search in conjunction with KNN matting and FBS parameter optimisations once after the initial training. The thresholds were optimised in 0.1 increments ranging from 0.1 to 0.4 for the lower bound and from 0.6 to 0.9 for the upper bound. The final thresholds used were 0.2 and 0.9.

5.4.2. Alpha-Matting

Alpha matting describes an image decomposition problem in which a pixel value in an image consists of two or more components such as foreground and background. Alpha matting is relevant for estimating the opacity of image components. The 2-component alpha matting is defined in (14). Alpha matting is by definition an ill-posed problem, and as such samples of known foreground and background pixels are given by the user. The samples are often given as a trimap, which indicates known foreground, known background, and unknown pixels.

KNN matting [48] is a nonlocal alpha matting method that solves the image layer alpha by computing K nearest neighbour (KNN) in feature space. KNN matting has four main parameters that should be adjusted to achieve optimal results on certain types of scenes. These parameters are lamda, level and, 2 parameters for K. Lamda denotes the confidence that the user has in matting annotations, level denotes spatial coherence. K_1 denotes the number of neighbours with the level spatial coherence and K_2 denotes the number of neighbours with weak spatial coherence. The values for lambda and level are optimised with random search in this research, while K is left as the default $K_1 = 10$ and $K_2 = 2$. A publicly available Matlab implementation[7] of KNN matting [48] is used in this implementation.

5.4.3. Edge-Aware Smoothing

Edge-aware smoothing techniques exploit a priori in natural images that different image properties are contained within smooth boundaries and differ across discontinuity boundaries. Explained in terms of semantic segmentation, for example, this means that pixels in a uniform image area such as uniform colour are quite likely a part of the same semantic segment. This priori can be described as bilateral-smooth. Regions are smooth within boundaries and non-smooth across boundaries. This work uses Fast Bilateral Solver introduced by Barron and Poole in 2016 [47].

Fast Bilateral Solver optimises a minimisation problem that encourages the output to be bilateral-smooth by minimising the squared residual defined as

$$\min_x \frac{\lambda}{2} \sum_{i,j} \hat{W_{i,j}} (x_i - x_j)^2 + \sum_i c_i (x_i - t_i)^2 \tag{16}$$

[7] `https://github.com/dingzeyuli/knn-matting`

Parameter name	Final value	minimum value	maximum value	step size
KNN lambda	85	10	100	5
KNN level	3	1	15	2
Morphological circle kernel radius	5px	5	20	2
FBS sigma luma	4	4	32	4
FBS sigma chroma	8	4	32	4
FBS sigma spatial	8	4	32	4
Binary threshold	0.65	0.5	1.0	0.05
Trimap low	0.2	0.1	0.4	0.1
Trimap high	0.9	0.6	0.9	0.1

Table 1. Best performing results of parameter search.

where t is the target and c is confidence. $\hat{W_{i,j}}$ is an affinity matrix defined by Barron and Poole as

$$\hat{W_{i,j}} = exp(-\frac{|||[{p_i}^x, {p_i}^y]] - [[{p_j}^x, {p_j}^y]]||^2}{2{\sigma_{xy}}^2} - \frac{({p_i}^l]] - {p_j}^l)^2}{2{\sigma_l}^2} - \frac{|||[[{p_i}^u, {p_i}^v]] - [[{p_j}^u, {p_j}^v]]||^2}{2{\sigma_{uv}}^2}) \tag{17}$$

where σ_{xy}, σ_l and σ_{uv} are spatial, luma and chroma filtering parameters, and $p_{i,j}$ is a pixel in the filtering reference image and has spatial, luma and chroma information denoted as ${p_i}^x$, ${p_i}^y$, ${p_i}^l$, ${p_i}^u$, and ${p_i}^v$ respectively. Parameter values for σ_{xy}, σ_l, and σ_{uv} are optimised with random search in this research with the final values and search windows described in Table 1.

5.4.4. Parameter Optimisation

This implementation optimises a total of 9 parameters from the annotation generation pipeline using random search. The parameters are two trimap thresholds, KNN matting lambda and level, Fast Bilateral Solver's spatial, luma and chroma sigmas, kernel size for morphological operations, and final threshold for binarizing FBS outputs.

The parameter optimisation was implemented in Python as a random search over evaluation set 1 for 150 iterations and the parameters that maximised intersection-over-union for the image set were chosen for the whole duration of the annotation generation process. Evaluation set 1 is described in Chapter 6. More specifically, minimum and maximum ranges and step sizes for each parameter are defined. Random permutations of the parameter set are then iterated with repeating combinations excluded from the random selection. The mean intersection-over-union varied from 96.4% to 98.0% in the parameter search results. Segmentation accuracy on evaluation set 1 after initial training is at 93.5% intersection-over-union, so the additional processing described in Chapter 5 improves annotation accuracy by 4.5% on evaluation set 1 when using optimised parameters, and by 2.5% even when using non-ideal parameters. The best performing parameters had a precision of 98.98% and recall 99.03% while the worst performing set of parameters had a precision of 96.74% and recall 99.64%. The chosen parameters and their search windows are listed in Table 1.

5.5. Annotation Quality Control

Annotation quality control is implemented in conjunction with the annotation process. Generated annotations are labelled into three categories — good, fairly good, bad. This categorisation serves as the manual step in the annotation process and as simple quality control, trusting that the person labelling the images has a sound understanding of which annotations are good enough to be used in training. These images are added directly to the training set without further processing. Each image is rated by only one person, and while it could be argued that such quality control is not good enough for some use-cases, having more than one person annotate each image was considered unviable in this research.

Image categorisation is done in XnView MP[8], an image management software which allows binding hotkeys for rating an image from 1 to 5 and then moving to the next image in the directory. Such functionality is particularly interesting for this work as it minimises the time spent rating and then moving to the next image as only one press of a button completes both actions.

5.6. Manual Annotation Correction Tool

A Matlab implementation of a free-draw tool with options to assign pixels to either foreground or background label was implemented for this research. The next annotation is loaded to the screen automatically after an annotation is finalised. 5% of remaining image annotations are corrected manually in two iterations, and the corrected annotations are added to the training set without additional steps for quality control. Usually, manual annotations should go through a separate quality control step to minimise mistakes that might happen while drawing instance outlines. In this case, quality contol and manual annotation was done by the same person which may provide some extent of consistency in semantics, but it does not guarantee reliability or consistency of accuracy as shown in [24]. Overall the annotation process is treated as a specialist task in this research.

[8] https://www.xnview.com/en/xnviewmp/

6. EXPERIMENTAL EVALUATION

The main contribution of this work is to evaluate the specific workflow described in Chapter 5. To evaluate the results, a clear definition of the task and definition of evaluation metrics is required. Because the focus in this work is on one specific workflow, evaluation will be mostly focused on whether the workflow is viable for generating sufficiently high-quality annotations or not. Annotation accuracy is evaluated by directly comparing the generated annotations to manually annotated ground-truths with Jaccard index (11). The ideal target accuracy for this comparison is set to 96% as it was reported in [2, 43] that pixel-level annotations had an overlap of 96% between different annotators. However as the annotation category is subjective by definition in this research, it may not be reasonable to expect a 96% overlap for the task of main subject segmentation as main subject segmentation allows for large differences in ground-truths that may be equally valid. An example of such a case could be an image with one person closest to the camera and a group posing for the camera behind the person. An annotation that considers only the nearest person the main subject would be as valid as an annotation that considers both the nearest person and the group the main subjects. For this reason, the pixel-level overlap between generated and manually labelled annotations is also reported as a histogram. Figure 19 in Appendix 3 visualises this challenge in main subject segmentation.

The usefulness of the generated annotations is evaluated by reporting the performance of a semantic segmentation neural network on separate evaluation datasets that contain portrait image composition distributions that differ from the initial PortraitFCN+ dataset in scene modality. This comparison is done to evaluate whether the used workflow can extract useful information for a slight domain shift within the task of portrait segmentation.

Computational time is treated as a non-issue as all of the computation in this work can be scaled, and annotation time is solely focused on the amount of manual labour required to annotate a single image on average. Because time is also spent on classifying potential annotations as not good enough for training, the average annotation time is computed as the total time spent divided by the number of annotations categorised as good. Additionally, average annotation time is reported for manual correction using free-draw tool.

Results are also evaluated by training a smaller network on the generated data as well as with the manually annotated ground-truth data. It is relevant to note that not all of the 2010 images are annotated in this work. To match this, images that were not annotated are filtered out from the manually annotated ground-truth dataset. Segmentation accuracy is reported as the Jaccard index (11), and accuracy over the dataset is illustrated as a histogram. Improvement in performance is expected to be visible as a significant reduction in the number of images that are not segmented accurately by the network.

6.1. Segmentation Evaluation Datasets

Because a slight domain shift within the task of portrait segmentation is taken into account, five slightly different datasets are used for evaluation. All five datasets have

been annotated fully manually. Four of these datasets have been curated in-house and are not available publicly. These datasets are described in detail to provide a better understanding of the development of accuracy over iterations. The dataset shared by Shen et al. in PortraitFCN+ is used in addition to enable comparisons to existing work.

Evaluation set 1

Set 1 contains 393 images of similar modality to PortraitFCN+ data. While the images in this set have not been cropped based on face detection results unlike the images in PortraitFCN+, the dataset mostly consists of upper-body compositions.

Evaluation set 2

Set 2 contains 100 images that are exclusively full-body portrait images of scenes that have been found to be particularly challenging for the task. These scenes contain from one to five main subjects including uninvolved people in the background, challenging lighting conditions, and varying poses. This dataset contains the most significant change in modality in comparison to PortraitFCN+ dataset. An increase in accuracy on this dataset is assumed to display introducing new information to the training set. These images have been specifically captured for evaluating portrait segmentation and have been received from a customer who shall remain anonymous due to a non-disclosure agreement.

Evaluation set 3

Set 3 contains 1033 images collected from Flickr under the Creative Commons license. This set portrays a mixed modality of upper-body portrait images and full-body images in various lighting conditions and environments.

Ground-truth set

This set contains manually labelled annotations for all 2010 images in the dataset that is used for the process of generating annotations described in Chapter 5. The set contains various scenes, lighting conditions, and environments. Compositions range from upper-body portraits to full-body portraits, group photos, crowd photos, and upper-body portraits with a group posing in the background. Measuring the performance of the neural network in this dataset may not be particularly meaningful because it contains the same images that are used for training, although with different annotations. Regardless this dataset is used to train the network and compare the results to a network trained on generated annotations. All time-related evaluation is done on this dataset.

6.2. Annotation Time

Annotation speed is evaluated by recording the total time spent categorising the generated annotations on each iteration and computing the average annotation time per selected annotation. Additionally, two of the iterations introduce manually corrected annotations with the number of corrected annotations and total time spent correcting

Iteration	1	2	3	4	5	6	Total
Good	577	112	118	298	71	52	1228
Fair	401	760	631	645	621	547	3605
Bad	1032	561	572	192	145	126	2628
Total images reviewed	2010	1433	1321	1135	837	725	7461
Good percentage	28.7%	7.8%	8.9%	26.3%	8.5%	7.2%	16.45%
Total time	65min	73min	77min	37min	43min	27min	322min
Avg. per categorisation	1.94s	3.05s	3.5s	1.96s	3.08s	2.23s	2.59s
Avg. per good	6.76s	33.75s	39.15s	7.45s	36.34s	31.15s	15.73s
Manual corrections	-	-	68	-	41	-	109
Total correction time	-	-	123min	-	68min	-	191min
Avg. correction time	-	-	1min48s	-	1min40s	-	1min45s

Table 2. Annotation times and yields per iteration

annotations manually. Table 2 shows all time-related measurements in this research. The focus in evaluation is on the average time spent per annotation that was accepted into the training pool. In this work, these are the candidate annotations that were categorised as "Good". Annotations categorised as "Fair", or "Bad" are considered by-products or time-sinks in this research as they are fed into the algorithm again after each iteration until they are either corrected manually or are categorised as "Good". As shown in Table 2, the average time per annotation remains below 40 seconds on each iteration with four out of six iterations averaging above 30 seconds. The average time is assumed to be directly affected by how "easy" the input data is for the neural network, i.e. how similar the input data is in comparison to the training data. Iterations 1 and 4 yield significantly higher percentages than the remaining iterations. The training data on iteration 4 includes manual corrections done in iteration 3, which could explain the significant boost in performance. However similar boost in performance is not visible on iteration 6, which may be explained to some extent by the fact that iteration 5 introduces a smaller percentage of manual corrections relative to the total size of the training dataset. Investigating this difference in performance between iterations further is beyond the scope of this research. Overall the average time per annotation is at 15.73 seconds.

As a comparison, manually correcting annotations with free-draw tool took on average 1 minute 45 seconds per annotation in this work. Citing Lin et al. in [24] that graph-cut annotation was found to be roughly 2.8 times faster than free-draw, this would translate to roughly 37.5 seconds per annotation on average when using graph-cut. The annotation workflow introduced in this work remains roughly 2.3 times faster than graph-cut based on this analogy. It is noted that evaluating annotation and neural network accuracy would be required in order to evaluate the usefulness of this speed-up in comparison to annotation tools that are based on graph-cut. Comparison to other computationally-assisted workflows is left for future research. The overall usefulness of the annotation workflow introduced in this research is further evaluated in the subsequent sections.

6.3. Annotation Accuracy

In addition to comparing annotation time and neural network performance, it is also relevant to compare annotation accuracy to fully manually labelled annotations. The main subject segmentation task in this research is by definition ill-posed as each ground-truth is by definition subjective. Regardless, fully manually labelled annotations are treated as absolute ground-truths in this section. Annotations produced by the workflow introduced in this research are compared to fully manually labelled annotations purchased from a TDaaS provider in this section. Figure 18 visualises the overall intersection-over-union distribution on the completed dataset which includes 109 manually corrected annotations. As expected, the annotated dataset does not reach the benchmark of 96% intersection-over-union against the ground-truth dataset. The constructed dataset achieves an overlap of 94.79% mean intersection-over-union. Full visualisations for each iteration are shown in Appendix 2. Annotation accuracies remain within the range of 93.31% - 96.48%. Additionally figures 34 and 35 show the achieved mean intersection-over-union for manually performed annotation corrections at 94.80% and 93.55% respectively. Based on these measurements the introduced workflow achieves a respectable accuracy, although fully manual annotations should be tested for the task of main subject segmentation to draw conclusions on annotation overlap between repeated annotations. Manual corrections are also affected by the introduced workflow, which should be taken into account when evaluating the results.

A particular issue with the task of main subject segmentation is raised in Figure 19. Photograph in 19 is by Parklartatar and published at Flickr[9]. While the measured intersection-over-union is only 57.2%, both annotations can be considered roughly equally accurate, as annotators were instructed to label the people that they think are the main subjects in an image. Additionally to main subject segmentation in this research, annotators were instructed to label bottles, cups, and other items that subjects may be using. Based on the specifications the annotation produced by the workflow in Figure 6.19(b) could be more accurate as it also includes the bottle, but if quality control decides that both subjects should certainly be included in the annotation, then the annotation shown in Figure 6.19(c) can be considered significantly more accurate. Appendix 3. shows three additional annotations and differences to corresponding ground-truth annotations.

6.4. Neural Network Output Accuracy

Because the use-case is training neural networks that produce desired results, the usefulness of data is arguably best evaluated on segmentation network training results. If segmentation performance cannot be shown to improve, any improvements in annotation time would be rendered useless. Figure 20 shows the Jaccard Index development on evaluation sets described in the previous section. Iteration 0 represents the results achieved by training on the PortraitFCN+ training set published by Shen et al. Some improvements in performance can be seen on most evaluation sets with the most notable improvements on evaluation set 2 which differs the most from the

[9] `https://www.flickr.com/photos/parktatar/45713710041/`, licensed under CC BY 2.0

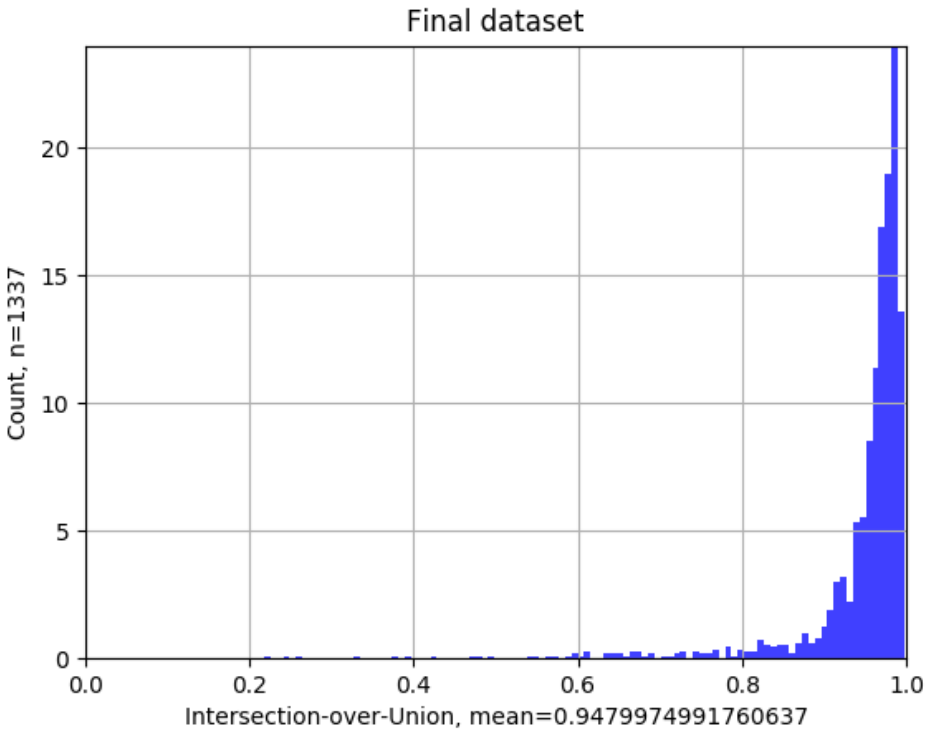


Figure 18. Intersection-over-Union of the created dataset in comparison to manually labelled ground-truth annotations.

initial training set in scene modality. It is also noted that the training data and annotation ground-truth sets are essentially the same, and as such accuracy can be expected to increase on this set. Ideally, the accuracy on annotation ground-truth set would approach 97% mean intersection-over-union. It also is important to note that iterations 3 and 5 include annotations that have been corrected manually. Overall, the segmentation performance is increased significantly. Interestingly the segmentation performance on the evaluation set 1 decreases after each semi-automatic iteration with notable improvements in performance on iterations 3 and 5. A more detailed look into this behaviour is outside the scope of this research. The performance on PortraitFCN+ evaluation set remained roughly the same with accuracy on the initial iteration at 97.38% mIoU and 97.49% mIoU on the final iteration. Full segmentation accuracy distributions are listed in Appendix 1.

(a) Image 2 (Photograph by Parklartatar published at Flickr `https://www.flickr.com/photos/parktatar/45713710041/`, licensed under CC BY 2.0).

(b) Segmentation of image.

(c) Ground-truth of image.

(d) Difference between ground-truth and generated annotation. Mean intersection-over-union 57.2%

Figure 19. Annotation visualisation.

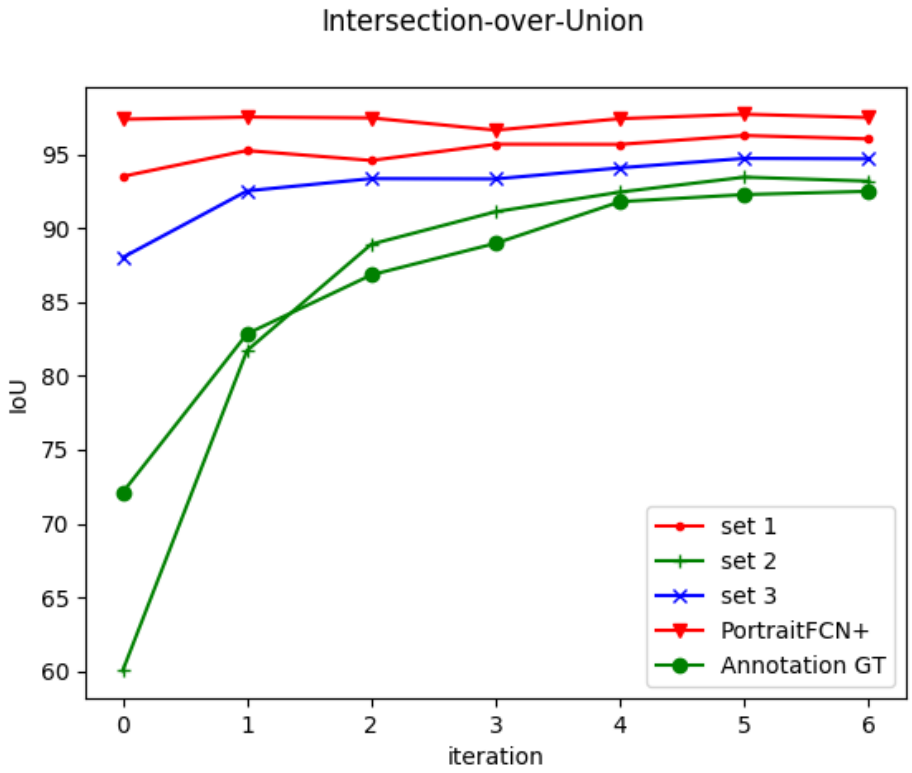


Figure 20. Neural network Intersection-over-Union on evaluation sets.

7. DISCUSSION

This work focuses on a specific workflow that has some resemblance to iterative workflows mentioned very briefly in existing research. While evaluation indicates that the introduced workflow works, a more comprehensive study with different types of tasks such as instance segmentation and different iterative workflows should be conducted for more conclusive results on iterative semi-automatic segmentation annotation. Particularly for more conclusive results, a less ambiguous segmentation task should be evaluated, as main subject segmentation is particularly ill-defined even for human annotators and quality controllers due to its' subjective nature. Additionally, a wider range of fully manual and computationally assisted labelling tools should be evaluated for a more comprehensive comparison in annotation time versus annotation accuracy.

The cost of creating a dataset is directly affected by how much labour it requires, and creating highly accurate datasets can be prohibitively expensive as workers must be compensated for their work accordingly. Faster workflows promise larger datasets, and typically larger datasets promise better results if the annotations are good enough. Quicker turnover-time is also an advantage in product development as it allows faster iteration and faster concept-to-product times. Therefore any improvements in annotation speed should be held in high regard as long as annotation accuracy does not suffer.

This research did not focus on quality control in the annotation process despite quality control being an important part of the process to ensure accurate annotations, as has been shown in previous research. Future work should evaluate how a separate quality control procedure affects the introduced workflow in comparison to other quality control procedures in terms of overall annotation accuracy.

The annotated dataset is rather small at only 1337 images including the manually corrected 109 annotations. The percentage of chosen annotations on each iteration ranged between 7.2% and 28.7% and a much larger number of iterations should be conducted to find out how the workflow performs over time and whether the efficiency of the workflow reduces to below the efficiency of manual or other computationally assisted annotation tools such as graph-cut, which already shows an increase in efficiency over free-draw annotation. Regardless, outside the scope of this research, the introduced workflow was used in a production environment to produce roughly 27,000 annotations in the task of main subject segmentation. These annotations were then used as part of a larger dataset to create a portrait image stylisation software solution that has been licensed to over 4 million mobile devices at the time of writing, and the workflow has been concluded to be capable of producing segmentation annotations that are good enough for use in production while significantly speeding up the process in comparison to free-draw annotations.

8. CONCLUSION

In this research focus was entirely on a specific workflow for semi-automatically generating semantic segmentation annotations, and evaluating whether the workflow can produce adequately accurate annotations for data that differs in scene modality from what was taught to the system initially. While brief mentions of similar workflows exist in literature, extensive evaluation of similar methods were not found in the initial literature review for this research. For this reason a publicly available dataset was selected for the initial training data to provide a point-of-reference for future research.

Evaluation of annotations shows that the introduced workflow produces annotations that have an intersection-over-union accuracy that is similar to manually corrected annotations while requiring significantly less manual labour than manual correction of annotations or fully manual annotation. Evaluation also indicates that the workflow can be used to extend the scene modality of a dataset beyond its' initial modality by significantly improving segmentation performance on on the chosen evaluation sets. Additionally outside the scope of this research, the introduced workflow has been used in a production environment to annotate roughly 27,000 images for the task of main subject segmentation, and the annotated data has been succesfully used to create a portrait image stylisation software solution that has been licensed to over 4 million mobile devices at the time of writing.

To conclude, the introduced workflow can produce annotations that are good enough for use in production while potentially not matching fully manual annotations in accuracy.

9. REFERENCES

[1] Kovashka A., Russakovsky O., Fei-Fei L. & Grauman K. (2016) Crowdsourcing in computer vision. Foundations and Trends® in Computer Graphics and Vision 10, pp. 177–243.

[2] Cordts M., Omran M., Ramos S., Rehfeld T., Enzweiler M., Benenson R., Franke U., Roth S. & Schiele B. (2016) The cityscapes dataset for semantic urban scene understanding. In: Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR).

[3] Mitchell T.M. (1997) Machine Learning. McGraw-Hill, Inc., USA, first ed., 2 p.

[4] McCulloch W.S. & Pitts W. (1943) A logical calculus of the idea immanent in nervous activity. Bulletin of Mathematical Biophysics 5, pp. 115–133.

[5] Rosenblatt F. (1958) The perceptron: A probabilistic model for information storage and organization in the brain. Psychological Review , pp. 65–386.

[6] Cybenko G. (1989) Approximation by superpositions of a sigmoidal function. vol. 2, pp. 303 – 314.

[7] Hornik K., Stinchcombe M. & White H. (1989) Multilayer feedforward networks are universal approximators. Neural Networks 2, pp. 359 – 366. URL: http://www.sciencedirect.com/science/article/pii/0893608089900208.

[8] Goodfellow I., Bengio Y. & Courville A. (2016) Deep Learning. MIT Press, 103-104 p. http://www.deeplearningbook.org.

[9] Linnainmaa S. (1970) Algoritmin kumulatiivinen pyöristysvirhe yksittäisten pyöristysvirheiden Taylor-kehitelmänä. Master's thesis, University of Helsinki.

[10] S. L. (1976) Taylor expansion of accumulated rounding error. BIT Numerical Mathematics 16, pp. 146–160.

[11] Schmidhuber J. (2015) Deep learning in neural networks: An overview. Neural Networks 61, pp. 85–117. Published online 2014; based on TR arXiv:1404.7828 [cs.NE].

[12] Rumelhart D.E., Hinton G.E. & Williams R.J. (1986) Learning representations by back-propagating errors. Nature , p. 533 536.

[13] Rumelhart D.E., Hinton G.E. & Williams R.J. (1986) Learning Internal Representations by Error Propagation, MIT Press, Cambridge, MA, USA. p. 318–362.

[14] Ruder S. (2016) An overview of gradient descent optimization algorithms. CoRR abs/1609.04747. URL: http://arxiv.org/abs/1609.04747.

[15] Aggarwal C.C. (2018) Neural Networks and Deep Learning. Springer, Cham, 497 p.

[16] Long J., Shelhamer E. & Darrell T. (2015) Fully convolutional networks for semantic segmentation. In: The IEEE Conference on Computer Vision and Pattern Recognition (CVPR).

[17] Ulku I. & Akagunduz E. (2019), A survey on deep learning-based architectures for semantic segmentation on 2d images.

[18] Ioffe S. & Szegedy C. (2015) Batch normalization: Accelerating deep network training by reducing internal covariate shift. CoRR abs/1502.03167. URL: http://arxiv.org/abs/1502.03167.

[19] Krizhevsky A., Sutskever I. & Hinton G. (2012) Imagenet classification with deep convolutional neural networks. Neural Information Processing Systems 25.

[20] Russakovsky O., Deng J., Su H., Krause J., Satheesh S., Ma S., Huang Z., Karpathy A., Khosla A., Bernstein M., Berg A.C. & Fei-Fei L. (2015) ImageNet Large Scale Visual Recognition Challenge. International Journal of Computer Vision (IJCV) 115, pp. 211–252.

[21] Sun C., Shrivastava A., Singh S. & Mulam H. (2017) Revisiting unreasonable effectiveness of data in deep learning era. pp. 843–852.

[22] Ronneberger O., Fischer P. & Brox T. (2015) U-net: Convolutional networks for biomedical image segmentation. CoRR abs/1505.04597. URL: http://arxiv.org/abs/1505.04597.

[23] Gonzalez R.C. & Woods R.E. (2006) Digital Image Processing (3rd Edition). Prentice-Hall, Inc., USA.

[24] Lin T.Y., Maire M., Belongie S., Hays J., Perona P., Ramanan D., Dollár P. & Zitnick C.L. (2014) Microsoft coco: Common objects in context. In: D. Fleet, T. Pajdla, B. Schiele & T. Tuytelaars (eds.) Computer Vision – ECCV 2014, Springer International Publishing, Cham, pp. 740–755.

[25] Visual object tagging tool (2020). https://github.com/microsoft/vott. Accessed 17.3.2020.

[26] Computer vision annotation tool. https://github.com/opencv/cvat. Accessed 18.3.2020.

[27] Mittal S., Tatarchenko M., Özgün Çiçek & Brox T. (2019), Parting with illusions about deep active learning.

[28] Matan O., Burges C.J.C., Le Cun Y. & Denker J.S. (1991) Multi-digit recognition using a space displacement neural network. In: Proceedings of the 4th International Conference on Neural Information Processing Systems, NIPS'91, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, p. 488–495.

[29] Chen L.C., Zhu Y., Papandreou G., Schroff F. & Adam H. (2018) Encoder-decoder with atrous separable convolution for semantic image segmentation. In: ECCV.

[30] Torralba A. & Efros A.A. (2011) Unbiased look at dataset bias. In: CVPR 2011, pp. 1521–1528.

[31] Everingham M., Van Gool L., Williams C.K.I., Winn J. & Zisserman A. (2010) The pascal visual object classes (voc) challenge. pp. 303 – 338.

[32] Everingham M., Van Gool L., Williams C.K.I., Winn J. & Zisserman A. (2006) The PASCAL Visual Object Classes Challenge 2007 (VOC2007) Results .

[33] Everingham m., Eslami S.M.A., Van Gool L., Williams C.K.I., Winn J. & Zisserman A. (2015) The pascal visual object classes challenge: A retrospective.

[34] Russell B.C., Torralba A., Murphy K.P. & T. F.W. (2008) Labelme: A database and web-based tool for image annotation. pp. 157 – 173.

[35] Shen X., Hertzmann A., Jia J., Paris S., Price B., Shechtman E. & Sachs I. (2016) Automatic portrait segmentation for image stylization. Computer Graphics Forum 35, pp. 93–102. URL: https://onlinelibrary.wiley.com/doi/abs/10.1111/cgf.12814.

[36] Park J.S., Chung M.S., Hwang S.B., Lee Y.S. & Har D.H. (2005) Technical report on semiautomatic segmentation using the adobe photoshop. vol. 18, pp. 333 – 343.

[37] Zhu B., Chen Y., Wang J., Liu S., Zhang B. & Tang M. (2017) Fast deep matting for portrait animation on mobile phone. In: Proceedings of the 25th ACM International Conference on Multimedia, MM '17, Association for Computing Machinery, New York, NY, USA, p. 297–305. URL: https://doi.org/10.1145/3123266.3123286.

[38] Visual object tagging tool (2020). https://vott.z22.web.core.windows.net/#/. Accessed 18.3.2020.

[39] Computer vision annotation tool (2019). Accessed 18.3.2020.

[40] Everingham M., Zisserman A., Williams C.K.I., Van Gool L., Allan M., Bishop C.M., Chapelle O., Dalal N., Deselaers T., Dorkó G., Duffner S., Eichhorn J., Farquhar J.D.R., Fritz M., Garcia C., Griffiths T., Jurie F., Keysers D., Koskela M., Laaksonen J., Larlus D., Leibe B., Meng H., Ney H., Schiele B., Schmid C., Seemann E., Shawe-Taylor J., Storkey A., Szedmak S., Triggs B., Ulusoy I., Viitaniemi V. & Zhang J. (2006) The 2005 pascal visual object classes challenge. In: J. Quiñonero-Candela, I. Dagan, B. Magnini & F. d'Alché Buc (eds.) Machine Learning Challenges. Evaluating Predictive Uncertainty, Visual Object Classification, and Recognising Tectual Entailment, Springer Berlin Heidelberg, Berlin, Heidelberg, pp. 117–176.

[41] Everingham M., Zisserman A., Williams C.K.I. & Van Gool L. (2006), The pascal visual object classes challenge 2006 results. http://www.pascal-network.org/challenges/VOC/voc2006/results.pdf.

[42] Mottaghi R., Chen X., Liu X., Cho N.G., Lee S.W., Fidler S., Urtasun R. & Yuille A. (2014) The role of context for object detection and semantic segmentation in the wild. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR).

[43] Caesar H., Uijlings J. & Ferrari V. (2018) Coco-stuff: Thing and stuff classes in context. In: The IEEE Conference on Computer Vision and Pattern Recognition (CVPR).

[44] Wadhwa N., Garg R., Jacobs D.E., Feldman B.E., Kanazawa N., Carroll R., Movshovitz-Attias Y., Barron J.T., Pritch Y. & Levoy M. (2018) Synthetic depth-of-field with a single-camera mobile phone. ACM Trans. Graph. 37. URL: https://doi.org/10.1145/3197517.3201329.

[45] Park H., Sjösund L., Yoo Y. & Kwak N. (2019) Extremec3net: Extreme lightweight portrait segmentation networks using advanced c3-modules .

[46] (accessed 23.2.2020), Releasing "supervisely person" dataset for teaching machines to segment humans. URL: https://hackernoon.com/releasing-supervisely-person-dataset-for-teaching-machines-to-segment-humans-1f1fc1f28469/.

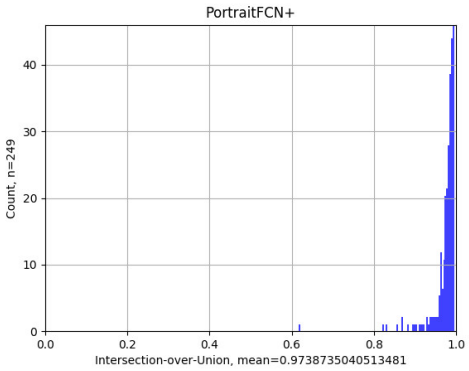
[47] Barron J.T. & Poole B. (2016) The fast bilateral solver. In: B. Leibe, J. Matas, N. Sebe & M. Welling (eds.) Computer Vision – ECCV 2016, Springer International Publishing, Cham, pp. 617–632.

[48] Chen Q., Li D. & Tang C. (2013) Knn matting. IEEE Transactions on Pattern Analysis and Machine Intelligence 35, pp. 2175–2188.

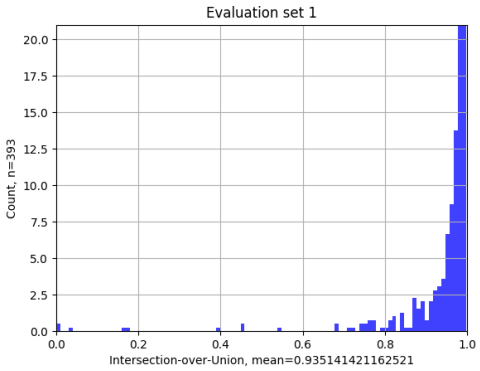
10. APPENDICES

Appendix 1 Jaccard Index histograms on each iteration (AI output accuracy)
Appendix 2 Jaccard Index histograms on each iteration (Annotation accuracy)
Appendix 3 Annotation visualisations

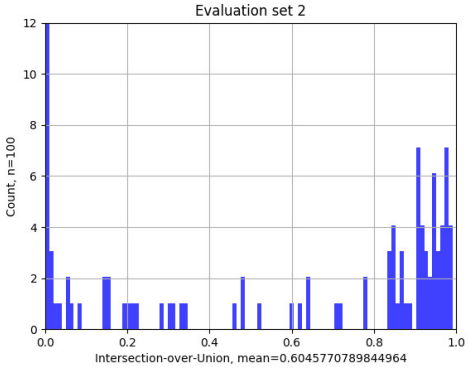
This appendix lists Deeplabv3+ xception 65 prediction accuracies on all listed evaluation sets trained on each iteration.



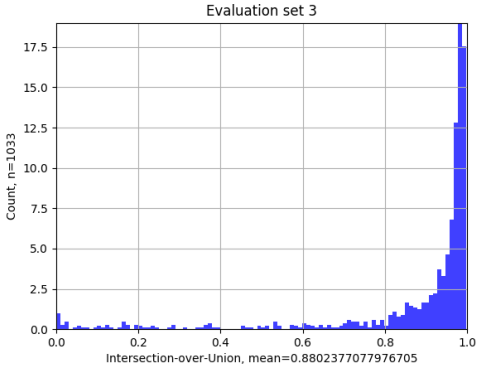
(a) PortraitFCN+



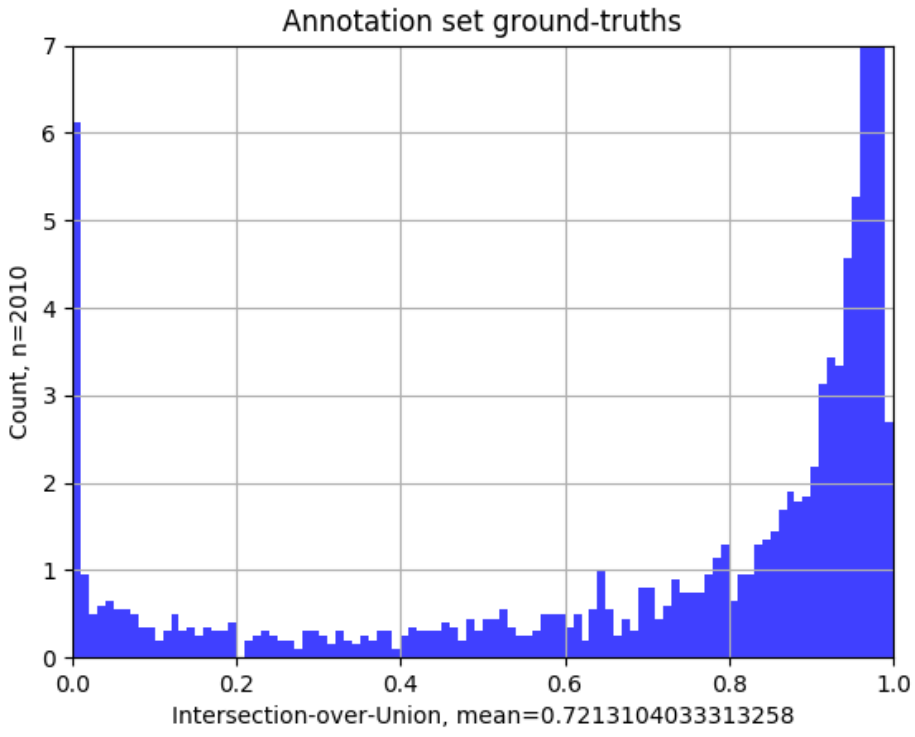
(b) Evaluation set 1



(c) Evaluation set 2

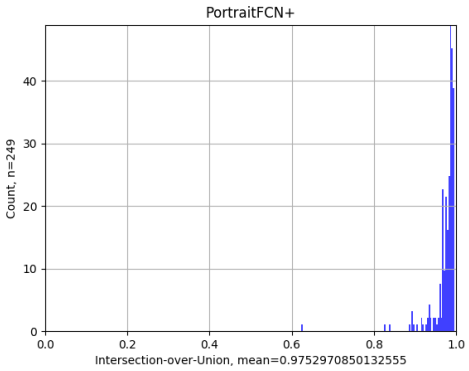


(d) Evaluation set 3

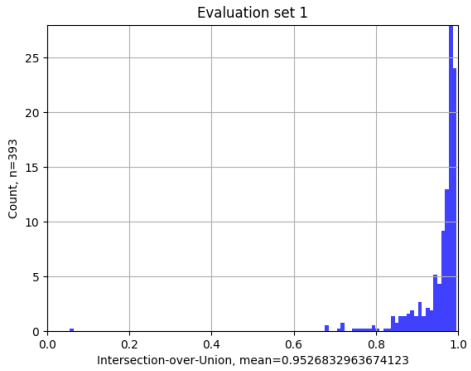


(e) Ground-Truth set

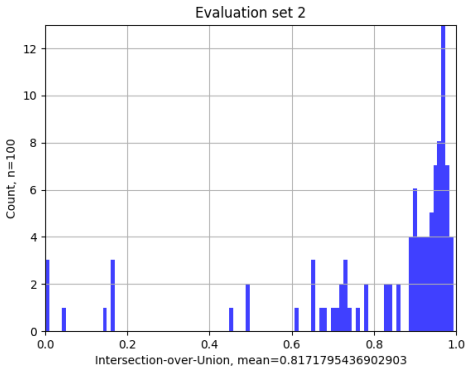
Figure 21. Initial accuracies trained on PortraitFCN+.



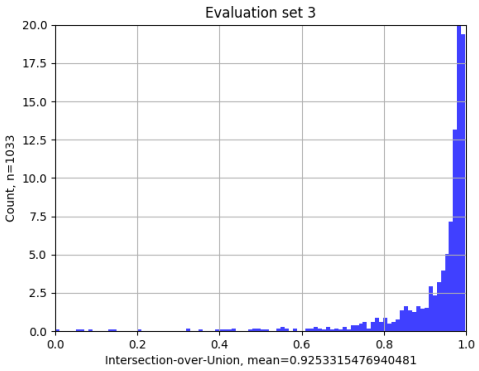
(a) PortraitFCN+



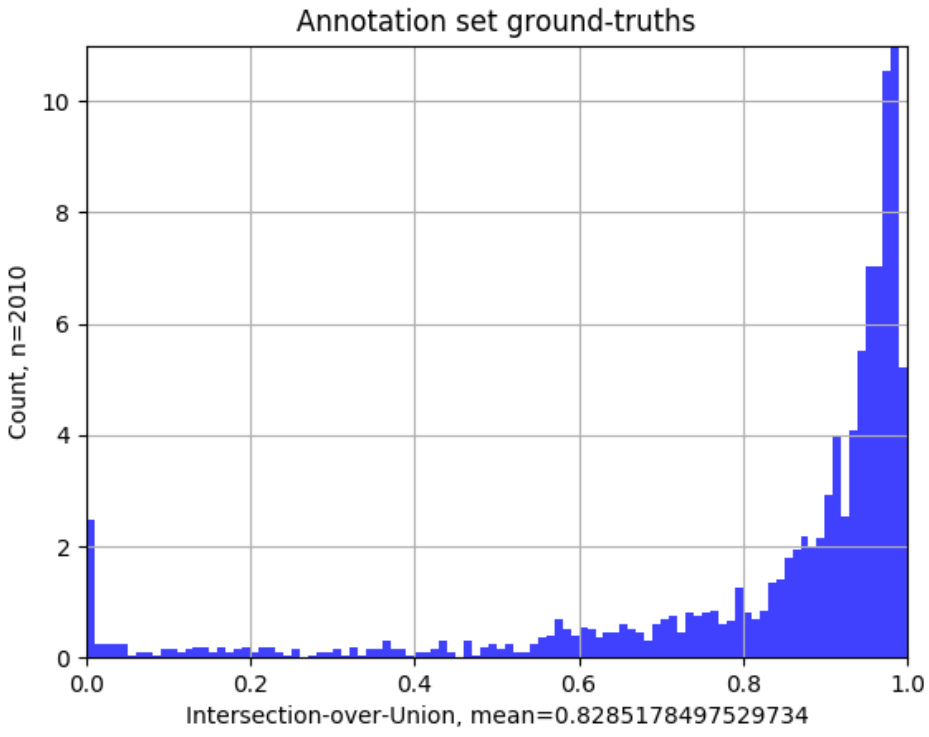
(b) Evaluation set 1



(c) Evaluation set 2

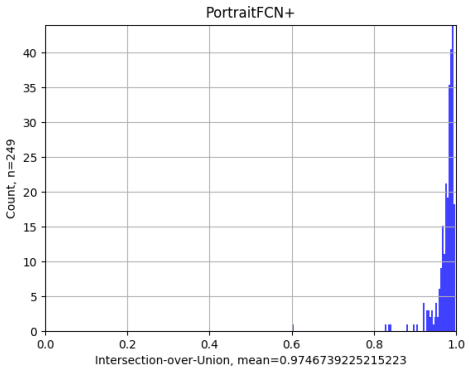


(d) Evaluation set 3

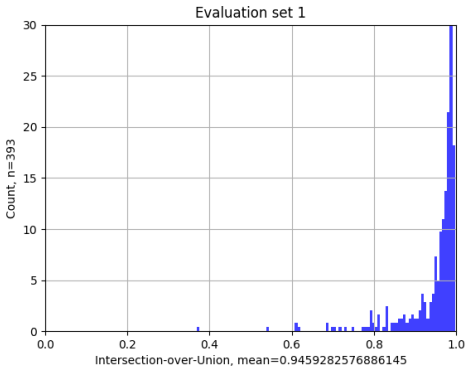


(e) Ground-Truth

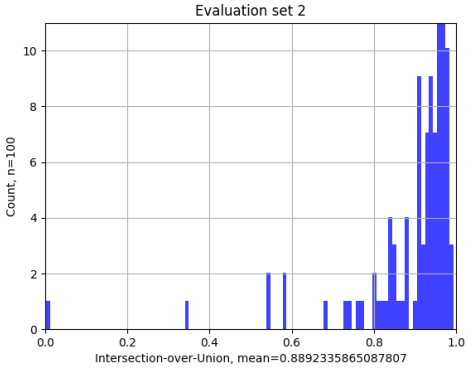
Figure 22. Accuracies trained on iteration 0.



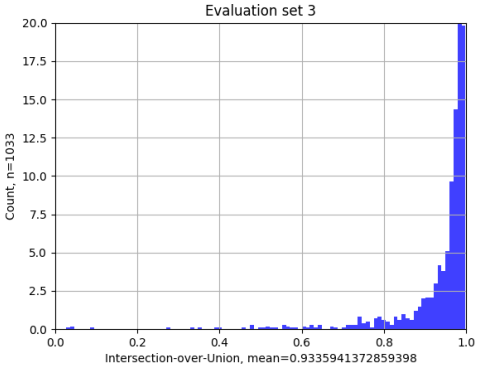
(a) PortraitFCN+



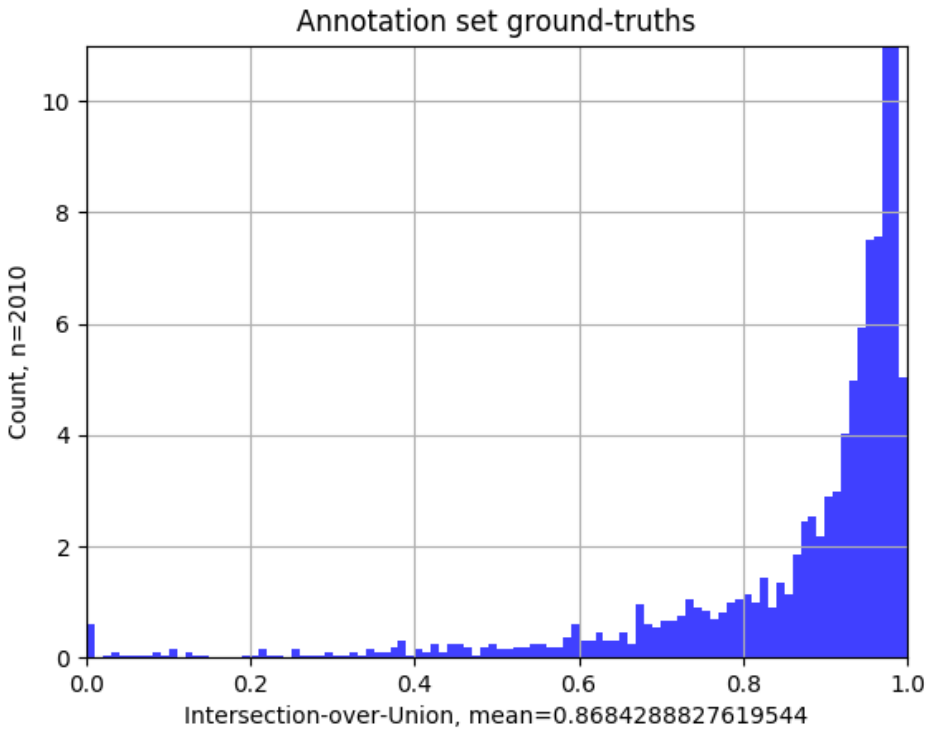
(b) Evaluation set 1



(c) Evaluation set 2

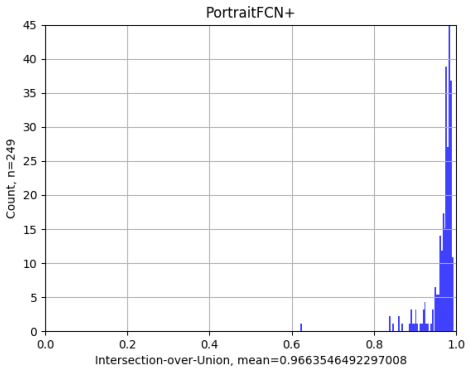


(d) Evaluation set 3

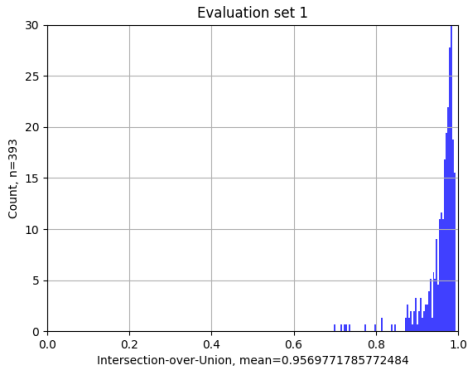


(e) Ground-Truth

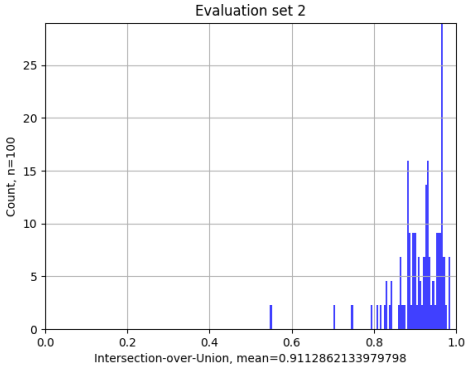
Figure 23. Accuracies trained on iteration 1.



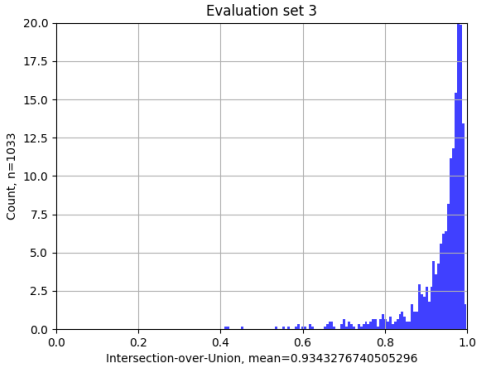
(a) PortraitFCN+



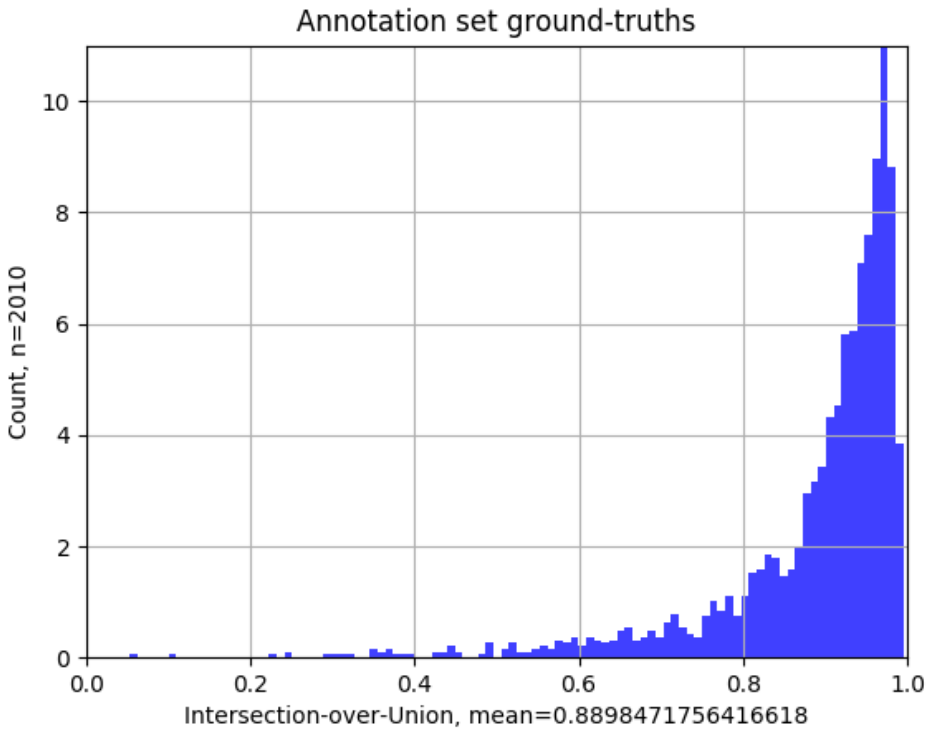
(b) Evaluation set 1



(c) Evaluation set 2

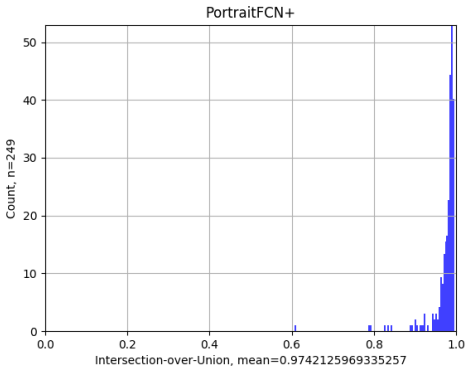


(d) Evaluation set 3

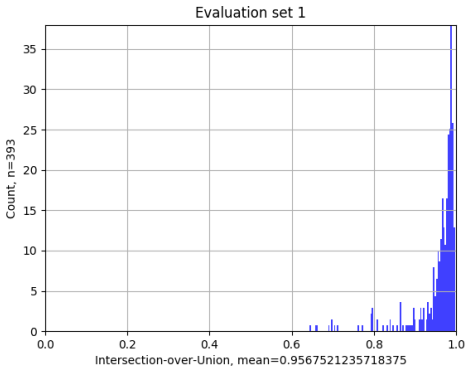


(e) Ground-Truth

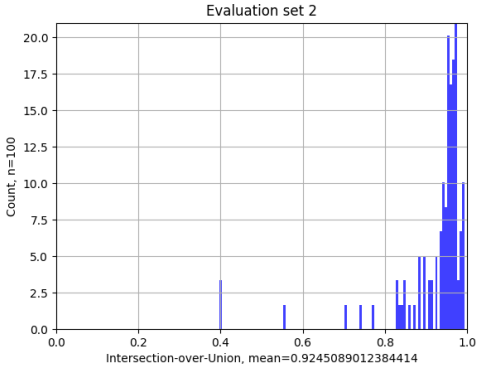
Figure 24. Accuracies trained on iteration 2 (Includes 68 manually corrected annotations).



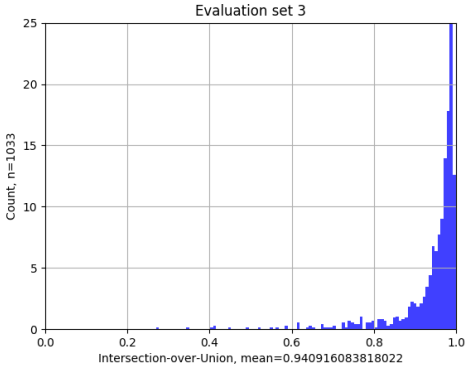
(a) PortraitFCN+



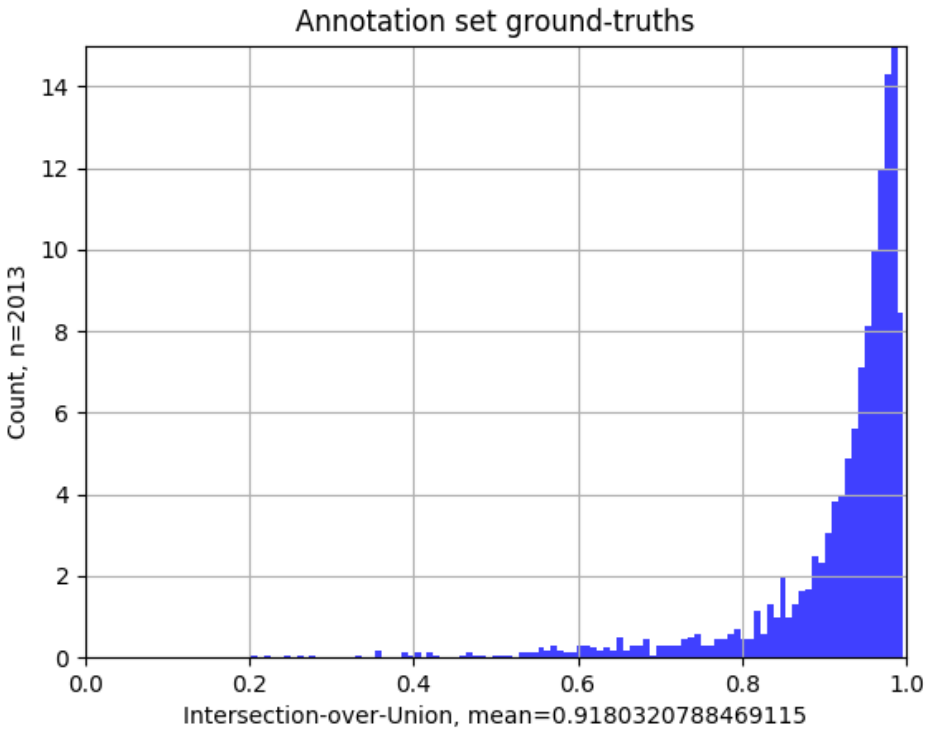
(b) Evaluation set 1



(c) Evaluation set 2

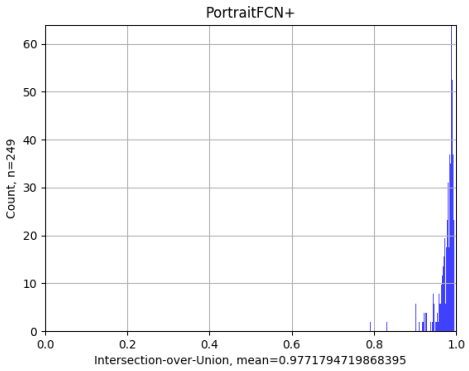


(d) Evaluation set 3

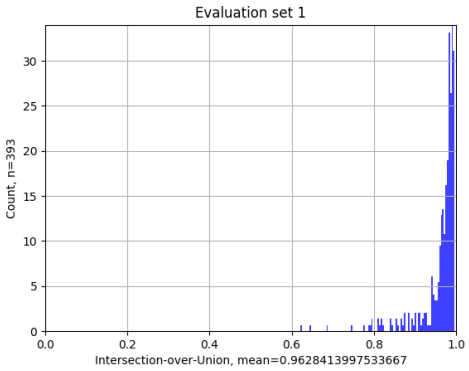


(e) Ground-Truth

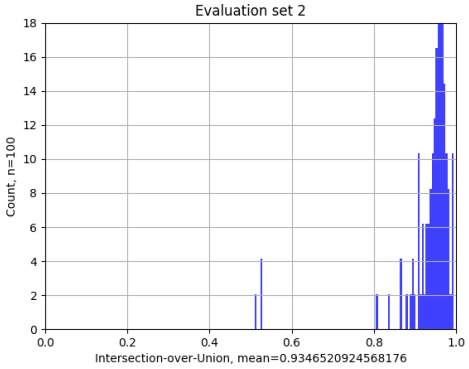
Figure 25. Accuracies trained on iteration 3.



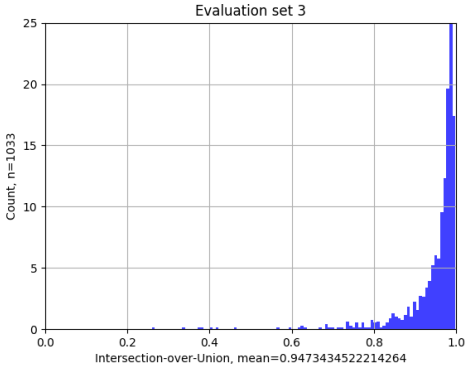
(a) PortraitFCN+



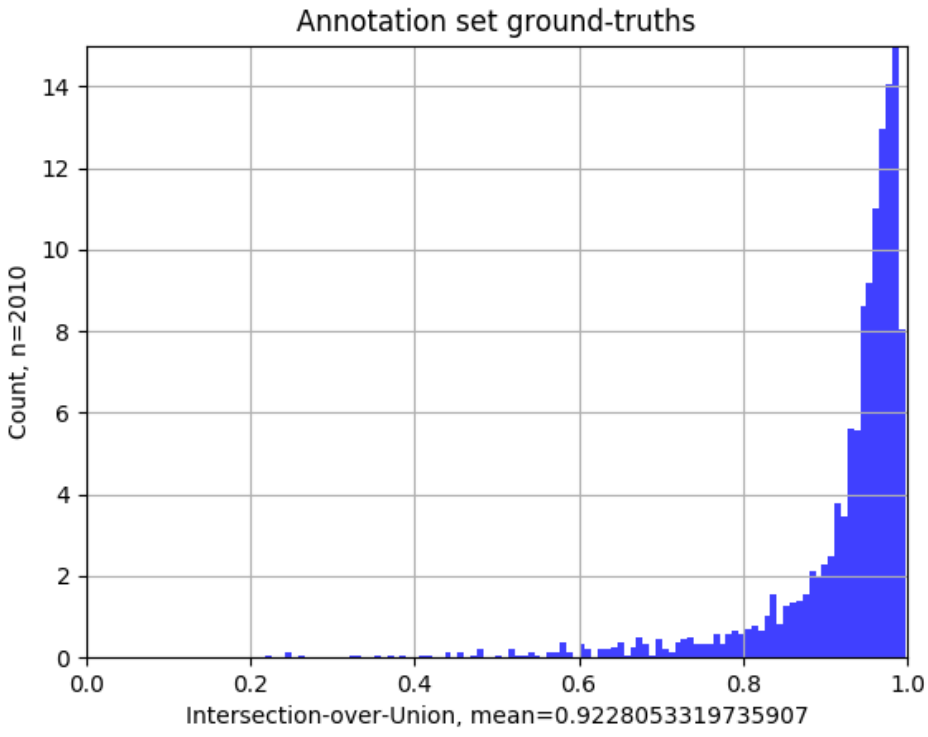
(b) Evaluation set 1



(c) Evaluation set 2

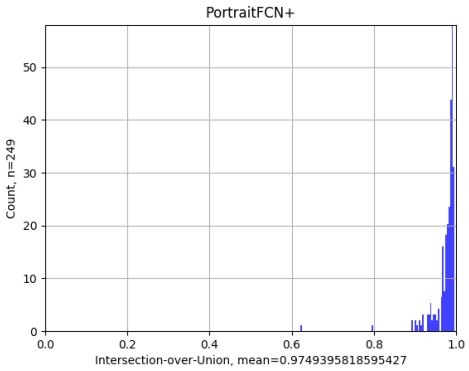


(d) Evaluation set 3

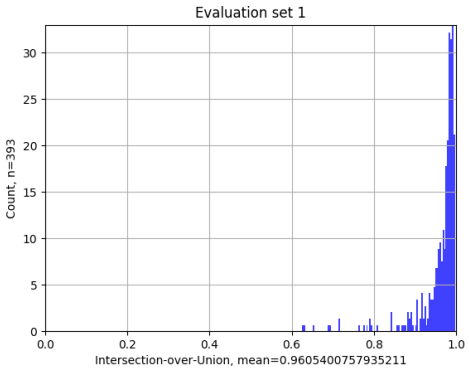


(e) Ground-Truth

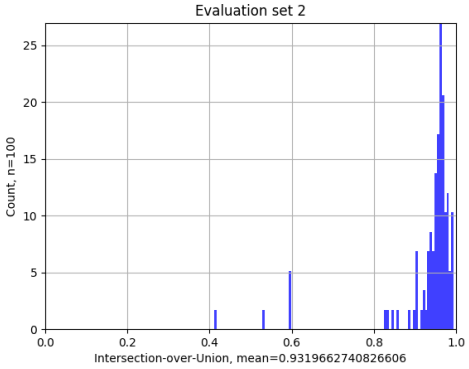
Figure 26. Accuracies trained on iteration 4 (includes 41 manually corrected annotations).



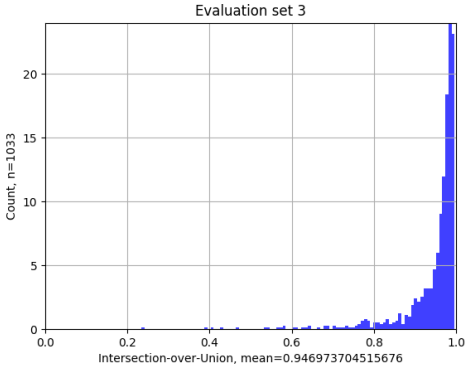
(a) PortraitFCN+



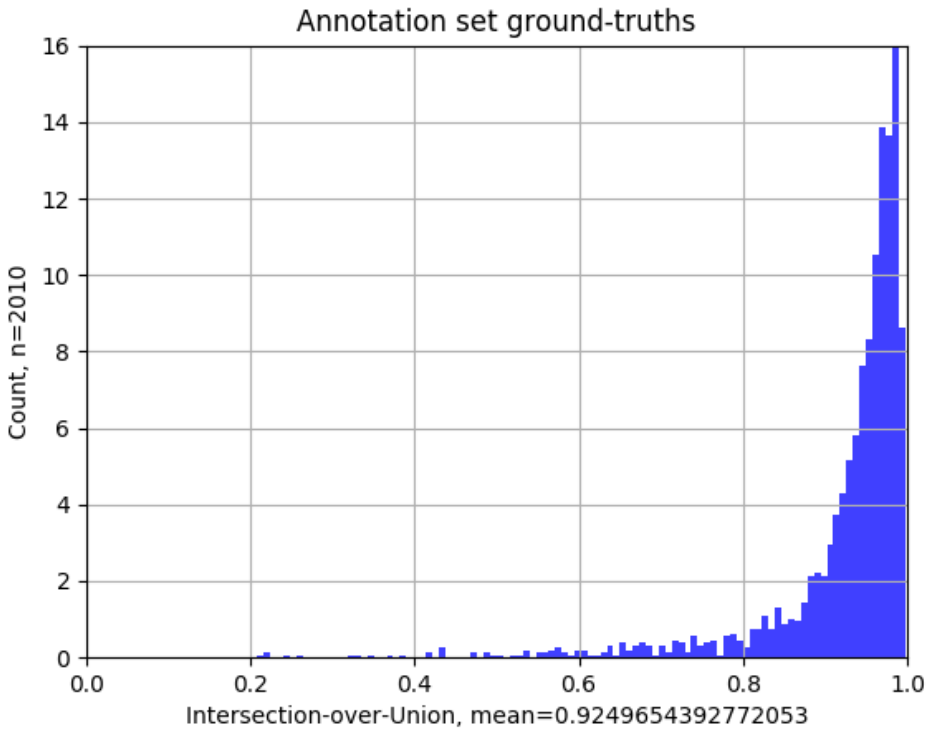
(b) Evaluation set 1



(c) Evaluation set 2



(d) Evaluation set 3



(e) Ground-Truth

Figure 27. Accuracies trained on iteration 5.

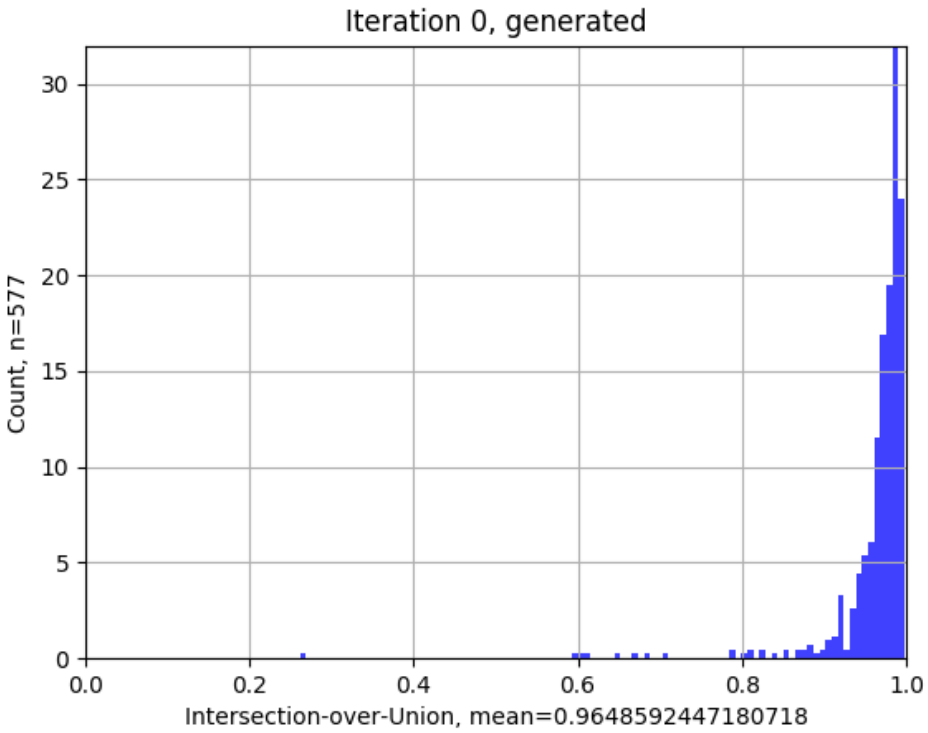


Figure 28. Intersection-over-Union on iteration 0.

Appendix 2 lists Jaccard index accuracies of each generated annotation on each iteration. Annotations generated by the workflow introduced in this work are compared against fully manually labelled annotations.

Subsequent figures illustrate the accuracy of manually corrected annotations in this work.

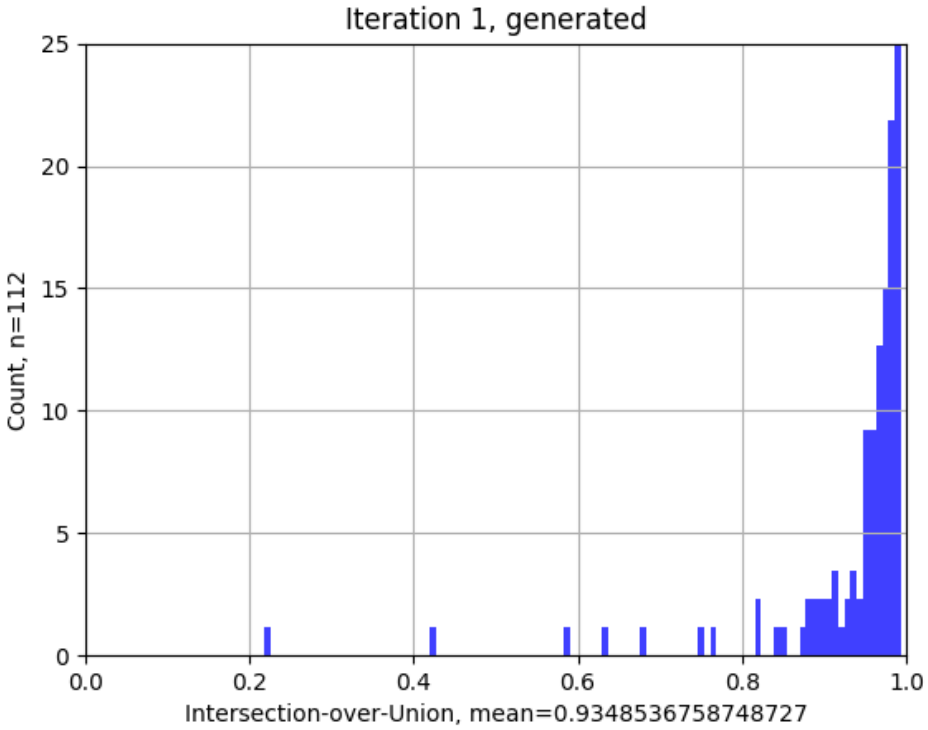


Figure 29. Intersection-over-Union on iteration 1.

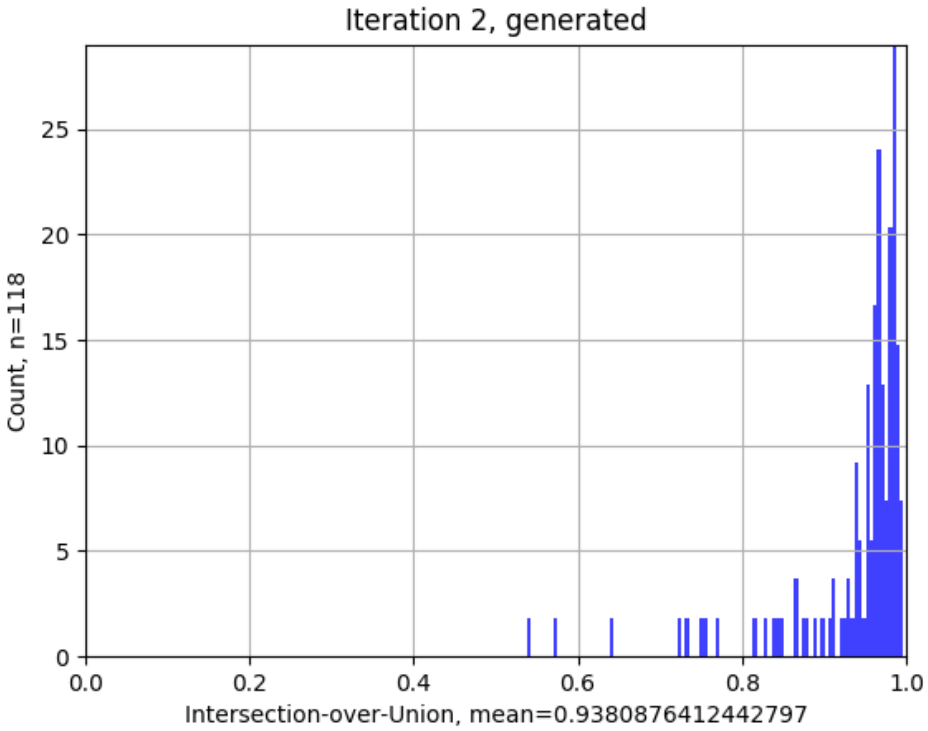


Figure 30. Intersection-over-Union on iteration 2.

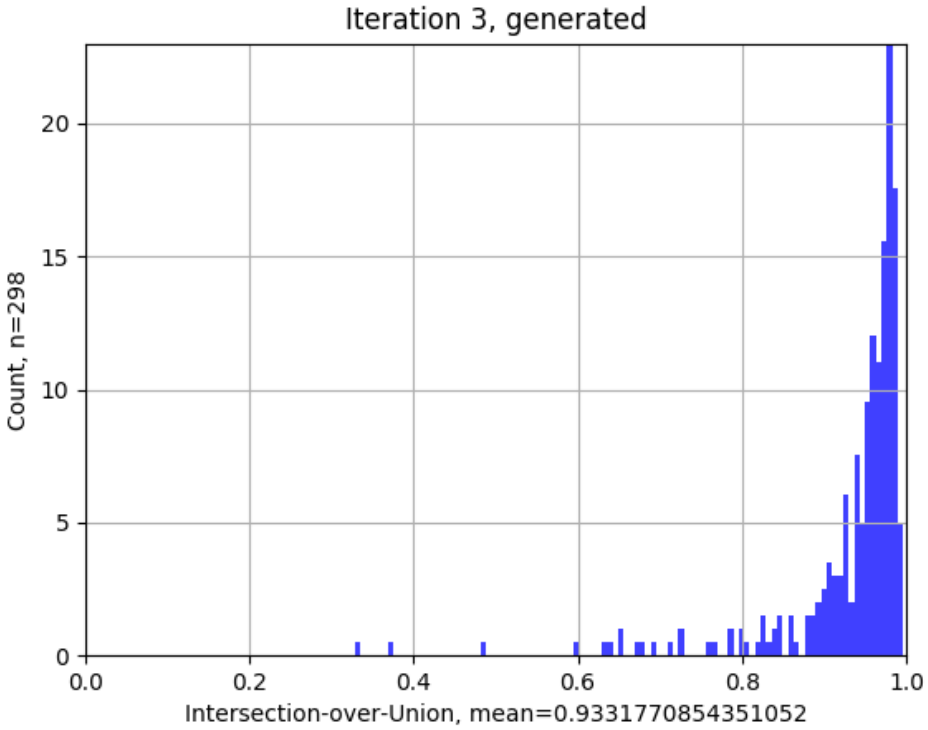


Figure 31. Intersection-over-Union on iteration 3.

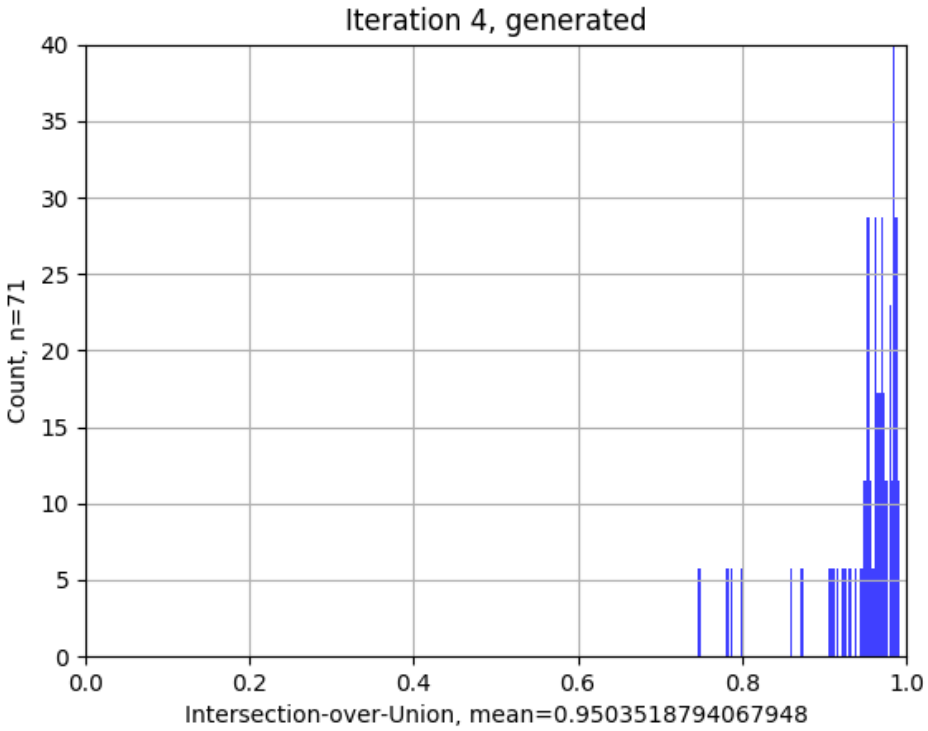


Figure 32. Intersection-over-Union on iteration 4.

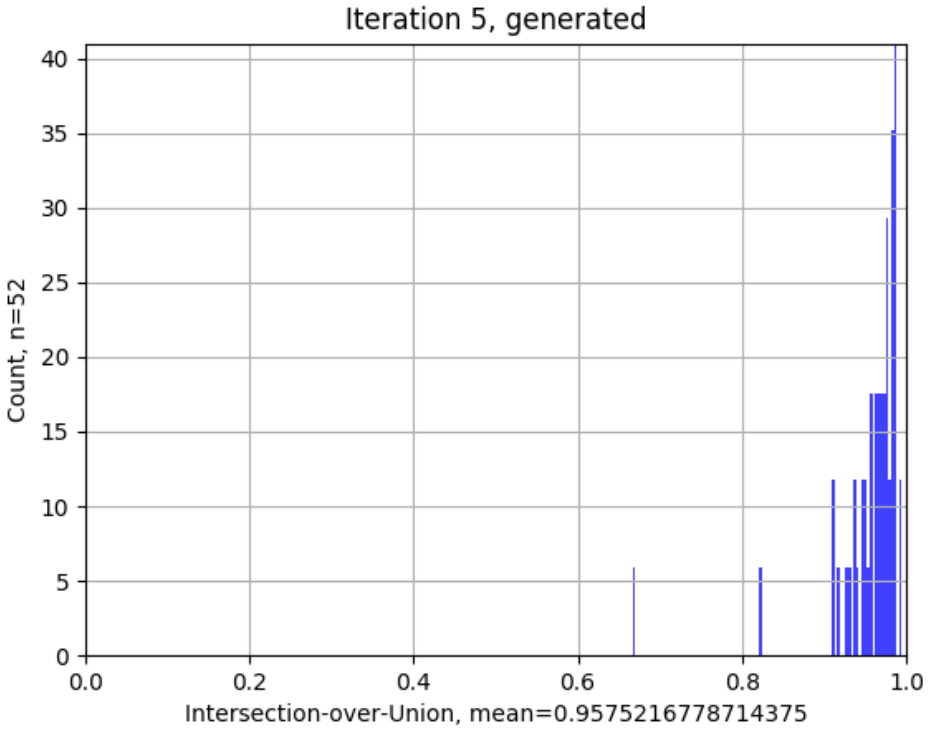


Figure 33. Intersection-over-Union on iteration 5.

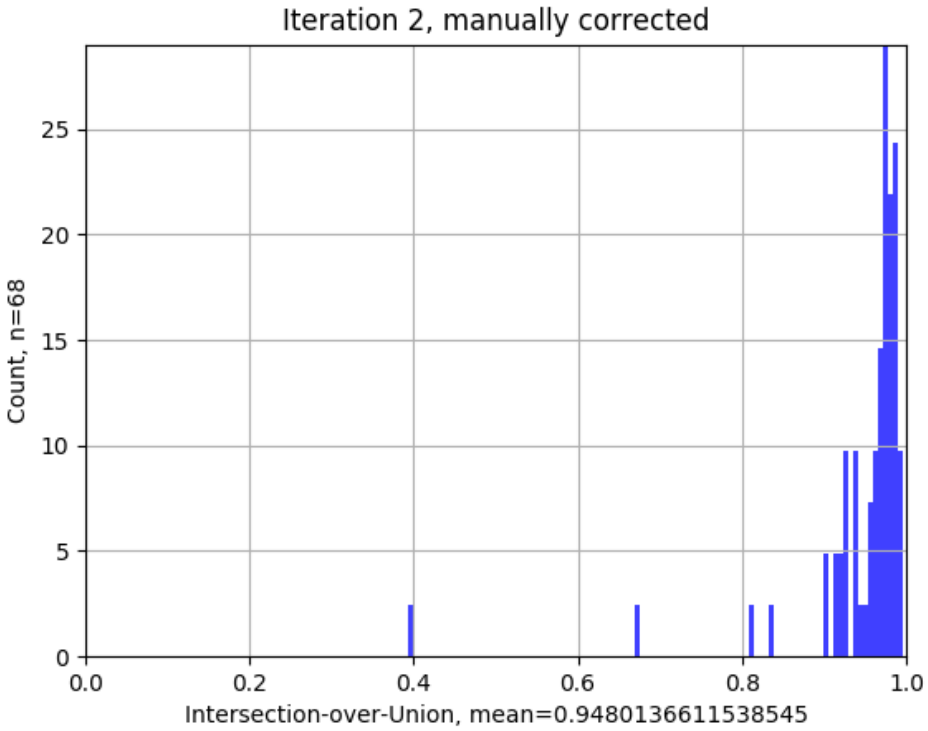


Figure 34. Intersection-over-Union of manually corrected annotations on iteration 2.

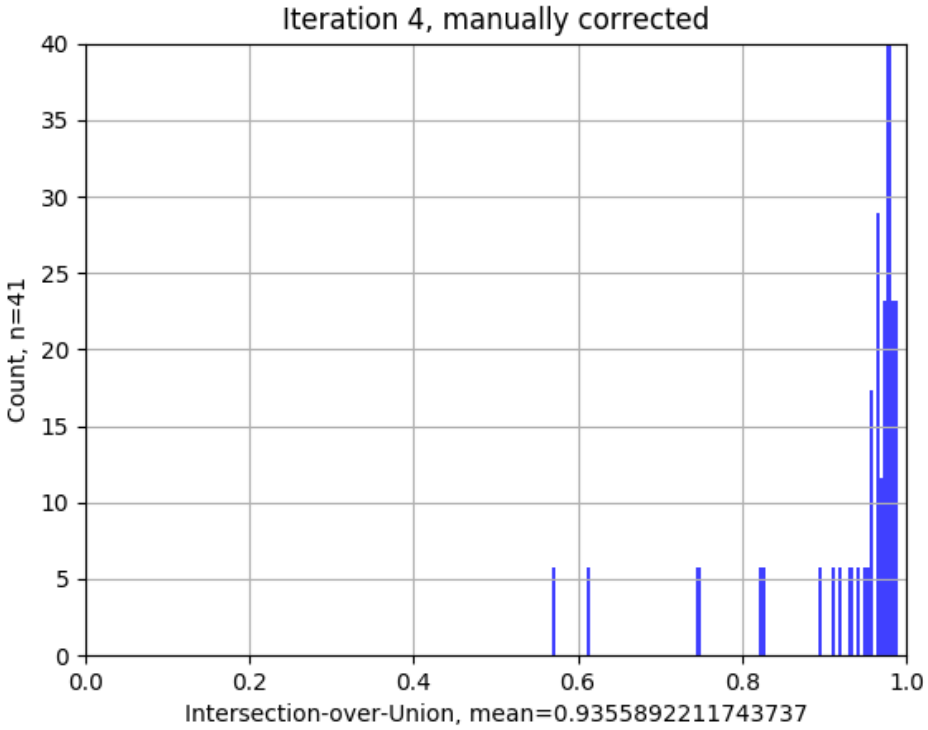


Figure 35. Intersection-over-Union of manually corrected annotations on iteration 4.

(a) Image 1.

(b) Segmentation of image 1.

(c) Ground-truth of image 1

(d) Difference between ground-truth and generated annotation. Mean intersection-over-union 89.7%

Figure 36. Annotation visualisation 1. Photograph by Ai ying sitan published at Flickr (https://www.flickr.com/photos/24702339@N08/31806041878/in/album-72157697226359240/, licensed under CC BY-SA 2.0).

(a) Image 2.

(b) Segmentation of image 2.

(c) Ground-truth of image 2.

(d) Difference between ground-truth and generated annotation. Mean intersection-over-union 66.9%

Figure 37. Annotation visualisation 2. Photograph by Ordiziako Jakintza Ikastola published at Flickr (`https://www.flickr.com/photos/jakintza_ikastola/43848049950/in/photostream/`, licensed under CC BY 2.0).

(a) Image 3.

(b) Segmentation of image 3.

(c) Ground-truth of image 3.

(d) Difference between ground-truth and generated annotation. Mean intersection-over-union 99.2%

Figure 38. Annotation visualisation 3. Photograph by Pawan Sharma published at unsplash (`https://unsplash.com/photos/lW2wFXv8JD4`, licensed under free license `https://unsplash.com/license`).