An Active and Transfer Learning Method for Instance Segmentation using Mask-RCNN

Jordan T. van Dijk
Jordan.van.Dijk@nl.ey.com

A thesis presented for the degree of
Master of Science

Universiteit Utrecht

Building a better working world

Business Informatics
Student number: 3980057
Utrecht University
Netherlands
11-11-2019

Contents

Abstract

State-of-the-art instance segmentation is one of the hottest topics in image recognition. Image recognition makes use of convolutional neural networks. Training convolutional neural networks require a vast amount of data and computational excelling machines. This is often not feasible for simple tasks. This master thesis investigates multiple machine learning methods to assist the user to label data and to train convolutional neural networks, without a need for large datasets and the newest computer. This master thesis proposes a tool that can train neural networks within two hours and gives promising results for small datasets.

Keywords: image recognition, convolutional neural networks, deep learning, active learning, transfer learning, instance segmentation

1. Introduction

Deep convolutional neural networks have led to a major increase in possibilities in the field of image recognition (Krizhevsky, Sutskever, & Hinton, 2012; Zeiler & Fergus, 2014; He, Zhang, Ren, & Sun, 2016; Shin et al., 2016). The improvement of performance in recent years is mainly due to the existence of larger labeled datasets, more powerful and deeper models, computational excelling machines and better techniques to prevent overfitting. In most real world situations there is a lack of labeled data samples (Otálora, Perdomo, González, & Müller, 2017). A vast amount of data and thus a large amount of variation is also needed to prevent overfitting. Three ways to reduce the pain of manually labeling data and to prevent overfitting are (i) data augmentation, (ii) active learning, and (iii) transfer learning.

This master thesis proposes a state-of-the-art tool to address these problems and combines the tasks of (i) obtaining data, (ii) transfer learning, (iii) annotating data, (iv) data augmentation, (v) instance segmentation using Mask RCNN, and (vi) probabilistic active learning. Data augmentation is used to improve the simplicity and speed of algorithms (van Dyk & Meng, 2001). It also increases the variation of the data and reduces the chance of overfitting. Active learning reduces the amount of manually labeled data, by querying the oracle (i.e. a human annotator) to label data that is most likely to improve the algorithm (X. Li & Guo, 2013). Transfer learning addresses the problem of needing large datasets to train a neural network, by using information gained from training a source domain (Pan & Yang, 2010). There do not exist any similar tools that try to combine all these different methods in one simple interface. Besides, tools made for image recognition are often either too simple or too complex. Too simple in the way that the neural network can not be trained on new data or too complex in the way that non-programmers cannot use it. The proposed tool is easy-to-use and can be adjusted to the needs of the user. This model can be trained in under two hours and gives good results when using a very small number of images.

Together with the tool, this master thesis proposes a methodology to test how well the different aspects of the tool perform. Transfer learning is essential when using a small dataset and improves the neural network predictions significantly. Probabilistic active learning and data augmentation in combination with transfer learning give similar results to just applying transfer learning. The methods are compared based on an average precision score given for five different Intersection over Union values. Furthermore, the methods are compared based on twelve different loss values. The methods are tested on two dataset and obtains a maximum average precision of 94%. This was done using only 45 images to train the Mask RCNN model.

This master thesis is structured as follows: In the next section, background information on neural networks, data augmentation, active learning and transfer learning is given. In section 3, the related work is discussed. Section 4, gives in depth information on the different methods discussed in section 2. In section 5, the methodology is described. Furthermore, it gives the configurations that are used to test the different methods. In section 6, a presentation and visualization of the results is given. It explains which methods should be used and the different possibilities a user of the tool has. In the final section the results will be discussed and future work will be stated.

2. Background

Image recognition is a vital thread in computer vision research (Vijayanarasimhan & Grauman, 2009). Deep convolutional neural networks have led to a major increase in possibilities in the field of image recognition (Krizhevsky, Sutskever, & Hinton, 2012; Zeiler & Fergus, 2014; He, Zhang, Ren, & Sun, 2016; Shin et al., 2016). Image recognition has four known fields: image classification, object detection, semantic segmentation and instance segmentation (Caelles et al., 2017; Michaelis, Ustyuzhaninov, Bethge, & Ecker, 2018). Image classification is used to classify and provide labels for a given image. Object detection is used to identify the object category and locate the position of an object using a bounding box (Ren, He, Girshick, & Sun, 2015). A bounding box is a minimal box boundary enclosing a set of interesting data points (i.e. an object) in an image. The idea behind semantic segmentation is to identify the object category of each pixel in an image (Long, Shelhamer, & Darrell, 2015). Instance segmentation is a combination of object detection and semantic segmentation (Romera-Paredes & Torr, 2016). Mask region-based Convolutional Neural Network (Mask RCNN) is a well-known method that achieves Instance Segmentation (He, Gkioxari, Dollár, & Girshick, 2017). This research uses the Mask RCNN framework and will be discussed in Section 5.

In the next paragraphs this master thesis will first explain neural networks and why they are used for image recognition. Then, three methods to reduce the amount of manual labeled data for training the neural network are discussed.

Neural networks. The last years have been crucial in the progress of training artificial neural networks that are matching and in some cases even surpassing human ability on various machine learning tasks (Gurney, 2014; Hannun et al., 2014; Taigman, Yang, Ranzato, & Wolf, 2014; Schroff, Kalenichenko, & Philbin, 2015; Yosinski, Clune, Nguyen, Fuchs, & Lipson, 2015). The best known types of neural networks are: (i) autoencoders, (ii) recurrent neural networks, and (iii) convolutional neural networks[1]. The first type of neural networks, autoencoders, are based on the observation that random initialization does not work great for training a neural network and that each layer should be pre-trained using an unsupervised learning method (J. Li, Luong, & Jurafsky, 2015). These type of neural networks have rarely been used in real applications. However, thanks to the upcoming of residual learning in recent years this framework has the potential to surpass the current performance on dimensionality reduction by other standardized methods (e.g. Principal Component Analysis (Section 4) and t-Distributed Stochastic Neighbor Embedding (Section 7) (He, Zhang, Ren, & Sun, 2016). The second type, recurrent neural networks are used on sequential data and can be used to predict with a certain probability (Graves, Mohamed, & Hinton, 2013). Recent breakthroughs are mainly in speech and text analysis (LeCun, Bengio, & Hinton, 2015). Another known field where this method is used is stock prediction. The final type is convolutional neural networks. They are primarily used to extract features from input images.

This master thesis will focus on convolutional neural networks since these have been proven to be successful for image recognition and object detection in many fields (Sener & Savarese, 2017). LeCun, Bengio, et al. (1995) states that: "The ability of multilayer back-propagation networks (i.e. convolutional neural networks) to learn complex, high-dimensional, non-linear mappings from large collections of examples makes them obvious candidates for image recognition." However, there exist other methods for image recognition and object detection like support vector machines and haar-like features. Support vector machines are a supervised learning model that uses data for classification and regression analysis[2] (Vapnik & Lerner, 1963; Boser, Guyon, & Vapnik, 1992; Drucker, Burges, Kaufman, Smola, & Vapnik, 1997; Drucker, Wu, & Vapnik, 1999; Chapelle, Haffner, & Vapnik, 1999; Chapelle & Vapnik, 2000). Chapelle, Haffner, & Vapnik (1999) shows that support vector machines can generalize well on image classification problems. In the past twenty years there have been a lot of improvements in techniques for neural networks in the field of image recognition and support vector machines are not used a lot anymore in this field[3]. Haar-like features are computational much faster than convolutional neural networks, but lack the ability to adapt to new situations[4]. For example, haar-like algorithms cannot recognize faces that are tilted or are covered partly. State-of-the-art methods therefore mainly use convolutional neural networks kernels. An image kernel is a small matrix that can be used in machine learning for feature extraction[5]. This method can recognize faces while covered or tilted, but is computational more costly. Besides, a vast amount of labeled data is needed to train these kind of models. In most real world situations there is a lack of labeled data samples (Otálora, Perdomo, González, & Müller, 2017). A vast amount of data and thus a large amount of variation is also needed to prevent overfitting. Three ways to reduce the pain of manually labeling data and to prevent overfitting are (i) data augmentation, (ii) active learning, and (iii) transfer learning.

Data augmentation. Data augmentation refers to a standard way of filling in missing data or as a solution for a lack of data (Tanner & Wong, 1987; Frühwirth-Schnatter, 1994). It refers to augmenting labeled data to create new data or to make the available data understandable. Data augmentation is used to improve the simplicity and speed of algorithms (van Dyk & Meng, 2001). The method was popularized in statistics for deterministic algorithms by Dempster, Laird, & Rubin (1977) and for stochastic algorithms by Tanner & Wong (1987). Deterministic algorithms always result in the same output given a fixed set of inputs and conditions. Stochastic models take uncertainty into account, which means that the output is

[1]`https://blog.statsbot.co/neural-networks-for-beginners-d99f2235efca`, accessed on 11-11-2019

[2]`https://medium.com/@dataturks/understanding-svms-for-image-classification-cf4f01232700`, accessed on 11-11-2019

[3]`https://medium.com/analytics-vidhya/the-scuffle-between-two-algorithms-neural-network-vs-support-vector-machine-16abe0eb4181`, accessed on 11-11-2019

[4]`https://towardsdatascience.com/whats-the-difference-between-haar-feature-classifiers-and-convolutional-neural-networks-ce6828343aeb`, accessed on 11-11-2019

[5]`http://setosa.io/ev/image-kernels/`, accessed on 11-11-2019

not unique given a fixed set of inputs and conditions. Most neural networks can be seen as a deterministic function when it is trained[6]. However, the process of training a neural network has stochastic elements (e.g. random initialization, nonlinearities and stochastic gradient descent). The idea of using data augmentation for image recognition comes from the fact that the performance of convolutional neural networks increases based on the amount of data it gets. Data augmentation is used to enlarge the training dataset size and helps to prevent overfitting by adding more variation to the data (Inoue, 2018). This is even if the data is of lower quality or similar to other images (Perez & Wang, 2017). The data augmentation used in this master thesis is stochastic and is used to create new data that is augmented randomly.

There exist many methods for data augmentation. This master thesis uses ten augmenting methods: (i) flip, (ii) rotate, (iii) scale, (iv) crop, (v) translation[7], (vi) lighting condition, (vii) perspective transformation[8], (viii) Gaussian Noise (Schlüter & Grill, 2015), (ix) Random Erasing (Zhong, Zheng, Kang, Li, & Yang, 2017), and (x) conditional generative adversarial nets transformation or style transfer (Gatys, Ecker, & Bethge, 2016). The first seven methods are basic augmentation methods and will not be discussed. However, the three final augmentation steps will be described. (viii) Gaussian Noise adds data points that have a probability density function equal to that of the normal distribution (i.e. Gaussian distribution) to all frequencies. This is done to make sure that high frequency patterns (which are often not part of the intended training data) are distorted so that the neural network will not only look at those. Of course this also means that lower frequency patterns are distorted too. However, a neural network can be trained to still use those lower frequency patterns. Another (less computational costly) known method that adds similar noise is the salt and pepper noise, which just spreads black and white pixels throughout the whole image. (ix) Random Erasing is done by randomly selecting a rectangle region and replacing its pixels with random values. (x) Conditional generative adversarial nets can transform an image from one domain to another domain. For example changing the theme of an image to a Picasso style theme. This method is very robust, but computationally too expensive for increasing the training set. A better alternative that is mainly used in data augmentation for increasing the size of the training set is neural style transfer. It mixes the ambiance of a reference image with the content of the image that needs augmentation. This produces an effect that is similar to conditional generative adversarial nets.

[6]`https://www.quora.com/Are-neural-networks-stochastic-or-deterministic`, accessed on 11-11-2019

[7]`https://medium.com/nanonets/how-to-use-deep-learning-when-you-have-limited-data-part-2-data-augmentation-c26971dc8ced`, accessed on 11-11-2019

[8]`https://medium.com/ymedialabs-innovation/data-augmentation-techniques-in-cnn-using-tensorflow-371ae43d5be9`, accessed on 11-11-2019

Active learning. The main idea behind active learning is that a machine learning algorithm can perform better when it can choose the data points that need to be labeled (Settles, 2010). Active learning reduces the amount of manually labeled data, by querying the oracle (i.e. a human annotator) to label data that is most likely to improve the algorithm (X. Li & Guo, 2013). Active learning generally consists of four components: (i) labeled training examples, (ii) unlabeled examples which labels can be obtained, (iii) something that can provide these correct labels, and (iv) a methodology to choose which unlabeled examples should be chosen (Holub, Perona, & Burl, 2008).

Several scenarios have been studied in the past in which the algorithm can use the oracle to learn: (i) membership query synthesis (Angluin, 1988), (ii) stream-based selective sampling (Atlas, Cohn, & Ladner, 1990; D. Cohn, Atlas, & Ladner, 1994), and (iii) pool-based sampling (Lewis & Gale, 1994; McCallumzy & Nigamy, 1998). The first, membership query synthesis, is not suitable for neural networks. This is due to the nature of how unlabeled instances are selected. The oracle can just create new instances instead of picking one out of the underlying distribution. To address this problem stream-based selective sampling and pool-based sampling were proposed. Stream-based selective sampling makes the assumption that obtaining an unlabeled instance is inexpensive. After obtaining the unlabeled instance the learner decides if it is useful to label this instance. To make this decision it uses query strategies to see if labeling the instance can result in increasing the learner. Pool-based sampling is commonly used in real-world problems. Pool-based sampling needs large collections of unlabeled data that is gathered once (Lewis & Gale, 1994). It makes the assumption that there is a small set of labeled instances from which it can learn a certain decision boundary. It asks the oracle to label instances that are likely to increase the accuracy of the learner. This last scenario of sampling is used in this master thesis.

Active learning can be applied to deep neural networks that are used for image recognition (Ducoffe & Precioso, 2018). The goal hereby is to minimize the amount of annotations required by the oracle to label the dataset. Annotations in instance segmentations can be seen as drawing polygons around the object to be labeled. Minimizing the amount of annotations can be done through (i) using a smart method of querying the right images to label, and (ii) pre-labeling new images from the unlabeled data pool.

First, this master thesis will look at how to query the right image to label. This can be done by using statistical methods (D. A. Cohn, Ghahramani, & Jordan, 1996). Some methods to do this are uncertainty sampling, Query-By-Committee (Ducoffe & Precioso, 2017), Expected Model change, Expected Error Reduction, Variance Reduction, Density-Weighted Methods (Settles, 2010), Deep Bayesian Active Learning (Gal, Islam, & Ghahramani, 2017) and (multi-class optimised) probabilistic active learning (Krempl, Kottke, & Spiliopoulou, 2014; Krempl, Kottke, & Lemaire, 2015; Kottke, Krempl, Lang, Teschner, & Spiliopoulou, 2016). This master thesis uses a probabilistic active learning method. Krempl, Kottke, & Spiliopoulou (2014); Krempl, Kottke, & Lemaire (2015); Kottke, Krempl, Lang, Teschner, & Spiliopoulou (2016) illustrate how probabilistic active learning performs and what the advantages are over other known methods.

Secondly, this master thesis will look at how the queried image can already be labeled by the machine when selected. This type of active learning is also known as cooperative machine learning (Wagner et al., 2018). To achieve this the output and predictions from the Mask-RCNN model are used in combination with the active learner. Hereby, it can be predicted if the image contains the to be found object and the position of the object. The model will use images where the label has a high confidence and gives it to the oracle. The oracle can then add, adjust or delete the polygons if needed. The principle is that at first the learner will make mistakes. But in time the performance of the learner will increase until a certain threshold is met. Once this threshold is met the learner can predict new objects in an image and add these to its training data, without interaction of the oracle.

Transfer learning. Many machine learning methods make the assumption that it can only work well if the training and test data are drawn from the same distribution and feature space (Pan & Yang, 2010). However, when the distribution or feature space changes the model usually has to be rebuilt from scratch. To rebuilt the model new training data is needed, which is often very time consuming and sometimes impossible in real world application. To address this problem transfer learning can be used. The key idea behind transfer learning is that learning something for one task can help you with learning a new task in the same domain or even in a different one (Taylor & Stone, 2009). For example, learning to play guitar, can help the understanding on how to play the piano. Applying knowledge learned on previous tasks can help with solving new problems faster or better.

There are two main approaches for transfer learning: (i) instance-based, and (ii) feature-based (Pan, Kwok, Yang, et al., 2008). The instance-based approach learns different weights to label training examples in the source domain to help learn to label in a target domain (Huang, Gretton, Borgwardt, Schölkopf, & Smola, 2007; Sugiyama, Nakajima, Kashima, Buenau, & Kawanabe, 2008). The feature-based approach tries to learn the shared feature structure of two domains to build a bridge for knowledge transfer between the two domains (Ando & Zhang, 2005; Blitzer, McDonald, & Pereira, 2006; Raina, Battle, Lee, Packer, & Ng, 2007). This master thesis focuses on an instance-based approach used on convolutional neural networks.

3. Related work

This section provides a brief review of previous work in image recognition and object detection. Since the introduction of convolutional neural networks by LeCun et al. (1989), it usage has grown immense (Zeiler & Fergus, 2014). From only being able to classify hand-written digits and detecting faces, recent years have shown that convolutional neural networks are capable of much more. The implementation of AlexNet (Krizhevsky, Sutskever, & Hinton, 2012) is the base of how convolutional neural networks are used currently. The improvement of performance in recent years is mainly due to the existence of larger labeled datasets, more powerful (deeper) models, computational excelling machines and better techniques to prevent overfitting. M. Lin, Chen, & Yan (2013) proposes a method "Network In Network" that makes use of 1x1 convolutional layers and implements micro neural nets globally in the neural network. This idea is used by Szegedy et al. (2015) to create GoogLenet (i.e. Inception net).

Simonyan & Zisserman (2014) illustrates the benefits of very deep nets in their work on VGGnet. However, deep neural nets have the problem of vanishing/exploding gradients, which hinders convergence (Bengio, Simard, Frasconi, et al., 1994; Glorot & Bengio, 2010). Shortcut connections with identity mapping are proposed by He, Zhang, Ren, & Sun (2016) in their work on residual neural network as a solution to this problem. Identity mapping is making a replica from the previous layer into a new layer. Szegedy, Ioffe, Vanhoucke, & Alemi (2017) combines the ideas of Inception networks and residual learning for even more powerful networks.

State-of-the-art methods that use these previously mentioned neural networks are MultiBox (Szegedy, Reed, Erhan, & Anguelov, 2014), Single Shot MultiBox Detector (SSD) (Liu et al., 2015), Faster Region-based Convolutional Neural Network (Faster-RCNN) (Girshick, Donahue, Darrell, & Malik, 2014; Girshick, 2015; Ren, He, Girshick, & Sun, 2015), Feature Pyramid Networks (FPN) (T. Lin et al., 2016), Region-based Fully Convolution Network (R-FCN) (Dai, Li, He, & Sun, 2016), You Only Look Ones (YOLO) (Redmon, Divvala, Girshick, & Farhadi, 2016), RetinaNet (T. Lin, Goyal, Girshick, He, & Dollár, 2017), and MobileNet (Howard et al., 2017). According to Huang et al. (2017) region based detectors like Faster-RCNN, FPN and R-FCN show a small accuracy advantage over single shot detectors like YOLO, SSD and MobileNet. However, this increase in accuracy can only be achieved when real-time speed is not needed[9]. Since speed of classifying an object is not the main objective for this research a method Mask-RCNN is used. Mask-RCNN is a simple to train model for instance segmentation and extends the Faster-RCNN method (He, Gkioxari, Dollár, & Girshick, 2017).

4. Preliminaries

In the following sections, this master thesis will dive deeper into how convolutional neural networks are structured. It will also look into weight initialization (which is important for transfer learning) and introduces the probabilistic active learning (PAL) method.

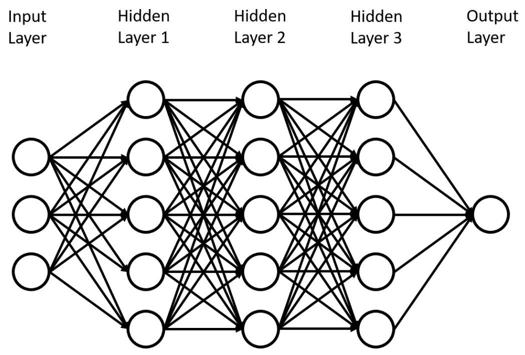


Figure 1: A fully connected neural network

Convolutional neural network. To introduce a convolutional neural network, first some general understanding of how a neural network is structured must be given. Figure 1 illustrates a fully connected neural network, where each neuron is connected to all the neurons of the previous network. A neural network is a network of nodes (i.e. neurons) and general has an input layer, one or more hidden layers and an output layer[10] [11]. A node in a neural network has (multiple) input and gives an output. Figure 2 visualizes one neuron. The neuron gets inputs $x_1, x_2, x_3, ..., x_n$ which are the outputs of a previous layer. Additionally, weights $w_1, w_2, w_3, ..., w_n$ are defined for every connection. The size of the weight indicate how important a connection is. Together with an activation function the output of the neuron is calculated. The output t, is determined by:

$$s = b + \sum_{i=1}^{n} w_i \cdot x_i \tag{1}$$

$$t = \sigma(s) \tag{2}$$

where b is a bias weight and σ the activation function.

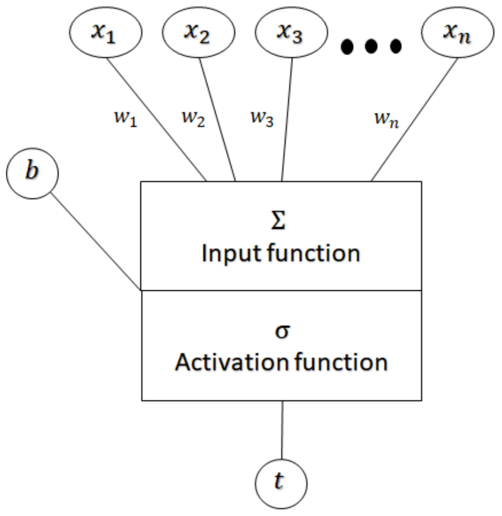


Figure 2: A single neuron

Looking at the number of neurons, connections and weights in a neural network a computational problem arises when using images as data. Each pixel of an image is a feature. Thus, resulting in a very high number of inputs. Given the fact that convolutional neural networks only connect one neuron with a small subset of neurons in the previous layer, make them more suitable than fully connected networks for image recognition[12]. A convolutional neural network is a more complex network and uses its neurons to find features that determine the classifications of images. Figure 3 shows a small convolutional network. The three important layers are (i) the convolutional layer (ii) the pooling layer and (iii) the fully connected layer.

The convolutional layer, which gives the network its name, is a linear operation that uses multiplication of the input with a set of weights. With images

[9]`https://medium.com/@jonathan_hui/object-detection-speed-and-accuracy-comparison-faster-r-cnn-r-fcn-ssd-and-yolo-5425656ae359`, accessed on 11-11-2019

[10]`http://cs231n.github.io/convolutional-networks/`, accessed on 11-11-2019

[11]`http://neuralnetworksanddeeplearning.com/chap1.html`, accessed on 11-11-2019

[12]`https://www.quora.com/What-is-the-difference-between-a-fully-connected-layer-and-a-fully-convolutional-layer`, accessed on 11-11-2019

[13]`https://towardsdatascience.com/understanding-neural-networks-from-neuron-to-rnn-cnn-and-deep-learning-cd88e90e0a90`, accessed on 11-11-2019

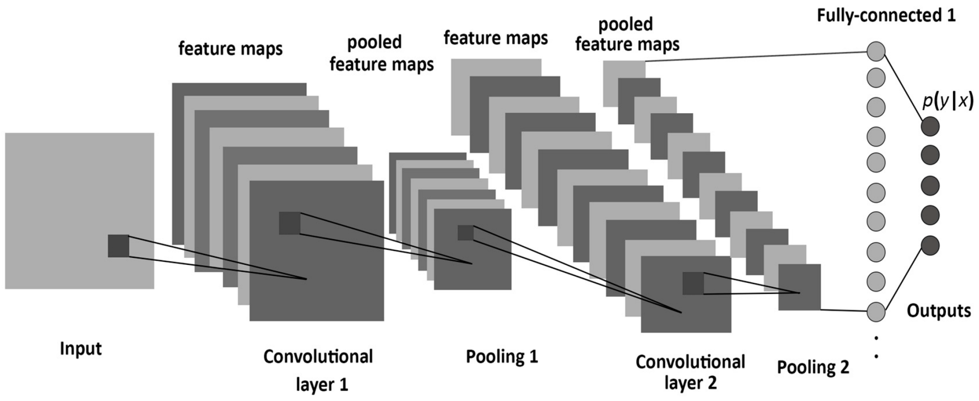


Figure 3: A simple convolutional neural network[13]

this can be seen as multiplying an input matrix W, with a matrix of weights, called a kernel or a filter K. This input matrix, is a MxN matrix, where M represents the vertical size of the pixels in an image and N the horizontal size. The kernel is smaller than the input data and slides over the whole image. This is used to detect a certain feature consistently in the entire image. Figure 4a shows how this is done using an input matrix of size $5x5$ and a weight matrix of $3x3$. However, sometimes the filter does not perfectly fit the input matrix. To address this problem zero padding P, can be added. This is also useful to make sure that features near the edge of an image are also taken into account. Another option to adjust how the filter moves is stride. Stride S, is the number of pixels that the filter shifts over the input matrix. Figure 4b illustrates padding and stride. The size of the output matrix (i.e. feature map) O, is determined by:

$$O = \frac{W - K + 2P}{S} + 1 \qquad (3)$$

In neural networks non-linear functions are used after a linear convolutional layer to introduce non-linearity to the activation map[14]. The three most popular non-linearity functions for neural networks are (i) Sigmoid, (ii) Tanh, and (iii) ReLU. The Sigmoid function takes the input and results in an ouput in the range of 0 and 1. It is given by $\sigma(k) = \frac{1}{1+e^{-k}}$, where k is the output of the convolutional layer. However, when the activation is at either tail, the gradiant becomes close to zero. In backpropagation this will result in a vanishing gradient. Tanh has outputs between -1 and 1. It has the same problem of saturation as the Sigmoid function, but outputs are zero centered. Rectified Linear Unit (ReLU) is probably the most popular function and computes $f(k) = \max(0, k)$. ReLU is more reliable and converges six times faster than Sigmoid and Tanh. Figure 5 visualizes these activation functions.

A convolutional layer is very useful to find the precise position of features in an image. However, a small rotation of features in an image will result in a to-

[14]`https://www.datascience.com/blog/convolutional-neural-network`, accessed on 11-11-2019

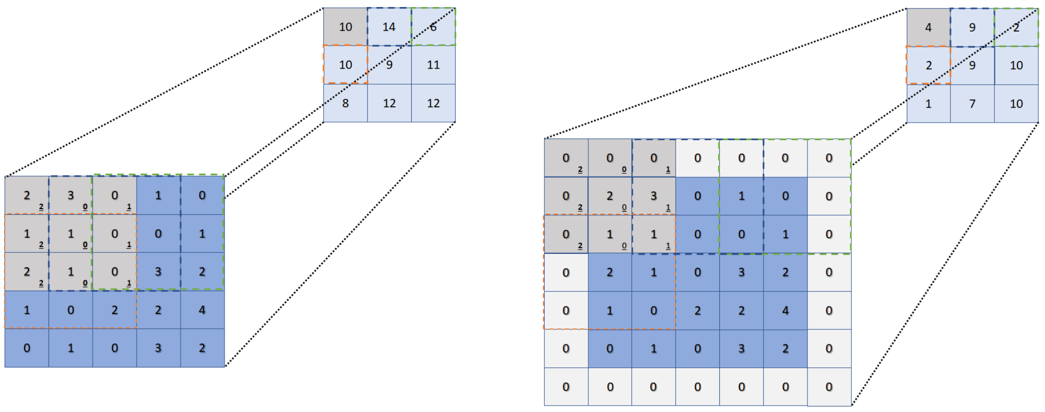


Figure 4: An example on how a 3 x 3 kernel moves over an image. (Left) a. Shows the movement of the kernel (gray) with P = 0 and S = 1 over an input matrix (dark blue) and gives an output (light blue). (Right) b. Shows the movement of the kernel (gray) over an input matrix (dark blue) with P = 1 and S = 2 and gives an output (light blue)

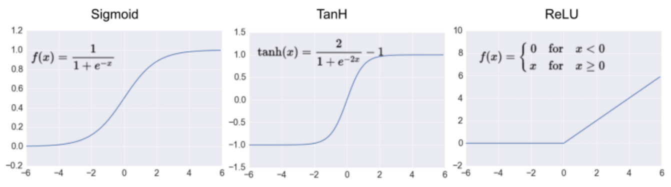


Figure 5: Activation functions

tally different feature map. To address this limitation, down sampling is used[15]. Down sampling can be achieved by using an extra convolutional layer. The more common and robust approach is to use a pooling layer. This layer is added after a non-linearity function. Pooling layers are used to reduce the number of parameters of each image in the dataset[16]. This reduction of the number of parameters can also be seen as creating a lower resolution version of an image. The two most common down sampling methods used for convolutional networks are (i) max pooling, and (ii) average pooling, see Figure 6. Similar to a convolution, pooling makes use of a filter to get an output. Often a $2x2$ filter with stride 2 is used to reduce the size of each feature map by 2. With max pooling the highest number is used of the feature map. Average pooling calculates the average of the feature map.

After (multiple) convolutional and pooling layers, a convolutional network ends with one or more fully connected layers. Neurons in the fully connected layers have full connectivity with all neurons in the preceding and succeeding layers, as can be seen in a standard neural network. The fully connected layer is used to combine the features found in the rest of the network. Finally, an activation function is used to classify the outputs[17]. Often a softmax activation function is used that turns the numbers from the neural network into probabilities[18]. These probabilities are used to give a certainty of the classification of objects in an image.

The depth of a network is of crucial importance and it was thought that adding more layers to a network should result in better classification (He, Zhang, Ren, & Sun, 2016). However, two problems arise when training deeper models: (i) vanishing/exploding gradients, and (ii) degradation. Vanishing/exploding gradients hampers the convergence of a neural network (Bengio, Simard, Frasconi, et al., 1994). This is caused by the slopes of the derivatives that become very small or very big. This problem is addressed by normalized initialization and intermediate normalization layers (Simard, LeCun, Denker, & Victorri, 1998; Glorot & Bengio, 2010; Saxe, McClelland, & Ganguli, 2013). This means that the weights are set accordingly to the input so that the output of each layer does not get too low or too high. Ioffe & Szegedy (2015) also addresses this problem by proposing batch normalization. The second problem, degradation, is the observation that adding more layers does not allow the network to learn as well as a smaller network (Srivastava, Greff, & Schmidhuber, 2015; He & Sun, 2015). This is counter intuitive, since it is assumed that deeper models are able to do at least as well as a simpler, shallower model. Take for example a deep network solution that copies all the layers from a smaller model and extends it by adding layers that are identity mapping. This should produce no higher training error, but experiments prove otherwise. This research uses a deep residual framework with 101 layers (ResNet 101), which uses shortcut connections with identity mapping to address the problem of the degradation problem (He, Zhang, Ren, & Sun, 2016). Shortcut connections are used to add input to the output after a few weight and activation function layers. Thus if the weight layers impose degradation, the model can always select the input from the shortcut connection. Figure 7 and 8 visualize the addition of a shortcut connection to a part of a neural network.

Besides the training algorithm of a neural network, weight initialization is the most important factor for the probability of convergence, the speed of convergence and the quality of predictions of the neural network (Fernández-Redondo & Hernández-Espinosa, 2001). Weight initialization is one of the most effective ways to speed up the training of a neural network (Drago & Ridella, 1992; Denoeux & Lengellé, 1993;

[15]`https://machinelearningmastery.com/pooling-layers-for-convolutional-neural-networks/`, accessed on 11-11-2019

[16]`https://www.datascience.com/blog/convolutional-neural-network`, accessed on 11-11-2019

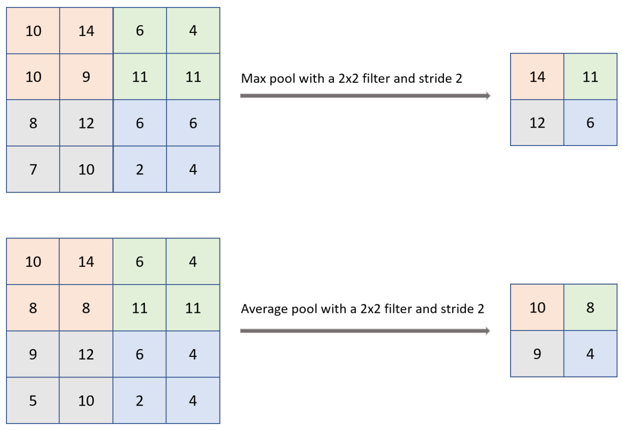


Figure 6: (Top) a. Max pooling,(Bottom) b. Average pooling

[17]`https://medium.com/@RaghavPrabhu/understanding-of-convolutional-neural-network-cnn-deep-learning-99760835f148`, accessed on 11-11-2019

[18]`https://medium.com/data-science-bootcamp/understand-the-softmax-function-in-minutes-f3a59641e86d`, accessed on 11-11-2019

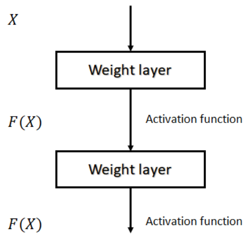


Figure 7: An abstract view of a normal neural network.

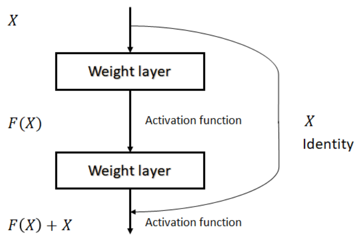


Figure 8: An abstract view of a residual neural network with a shortcut connection

Martens, 1996). For neural network a local optimization method is used for training and will result in a local optimum (Yam & Chow, 2000). Gradient descent is the go to local optimization technique for the training of neural networks. The local optimum found by gradient descent will determine the quality of the classification abilities of the neural network. When the local optimum is close or the same as the global optimum, the neural network will perform well. However, if the local optimum is not similar to the global optimum it will result in a poorly trained network. Besides gradient descent, a back propagation technique (Rumelhart, Hinton, & Williams, 1985; Rumelhart, Hinton, Williams, et al., 1988) is used to make sure that the error is optimized with respect to the weights (LeCun et al., 1989; Fernández-Redondo & Hernández-Espinosa, 2001). It is clear that the weight initialization has a large influence in the speed of convergence and influences if the training algorithm finds an acceptable local minimum.

There are multiple ways to initialize weights: (i) zero initialization, (ii) random initialization, and (iii) initialization through transfer learning. In zero initialization all the weights are initialized with 0[19]. If all the weights are initialized with 0, weights in subsequent iterations will keep the same value. This makes the hidden layers symmetric and does not make it better than a simple linear model. Assigning random values to the weights is a better method for neural networks, since it will result in different weights. However, a problem arises when values are very high or low. It will result in either vanishing or exploding gradients, which hampers convergence. He & Sun (2015) proposes an activation where the weights are initialized according to the ReLU activation function to resolve this problem. Glorot & Bengio (2010) proposes a similar method for TanH. The final method for weight initialization is through transfer learning (Raina, Battle, Lee, Packer, & Ng, 2007; Pan, Kwok, Yang, et al., 2008; Taylor & Stone, 2009; Pan & Yang, 2010; Shin et al., 2016). Transfer learning makes use of weights from a neural network that was trained for a different domain (i.e. source domain). These weights are trained to minimize the classification error of the different domain (i.e. target domain) (Ciresan, Meier, & Schmidhuber, 2012). The weights of the target domain are initialized using the final weights of the source domain. This is a very good method, especially when having a lack of data, since the weights are already trained on finding specific features. These specific features (e.g. edges, shapes, facial features) can be useful for finding the features in the neural network that needs to be trained.

Probabilistic active learning. There exist many active learning methods to help the annotator with querying images to be labeled. This master thesis will focus on probabilistic active learning (PAL), since it is a computational efficient and versatile method (Krempl, Kottke, & Spiliopoulou, 2014). PAL also shows promising results in the field of deep learning. There exist more optimized solutions like Optimised probabilistic active learning (OPAL) (Krempl, Kottke, & Lemaire, 2015) and multi-class OPAL (Kottke, Krempl, Lang, Teschner, & Spiliopoulou, 2016). These optimized solutions are cost-sensitive and non-myopic. Thus, may be capable in given better results. However, in the scope of this research these optimized solutions are too complex and are not used to show the benefits of using a probabilistic method for active learning. This master thesis also solely focuses on two classes and does not take multi-class scenarios into account.

The PAL method can be explained by three aspects: (i) label statistics, (ii) density weights, and (iii) probabilistic gains. Following the smoothness assumption (Chapelle, Scholkopf, & Zien, 2009), PAL considers that a candidate instance x, has the most influence on the classification in its neighborhood (Krempl et al., 2014). Thus, the effect of labeling a new candidate instance $(x, .)$, largely depends on the values of its neighboring labeled instances. This can be captured in the label statistics, $ls = (n, \hat{p})$, where n is the absolute number of instances in that neighborhood and $\hat{p}$ the number of positives. The values of n are obtained by counting the labeled instances, or approximated by

[19]`https://towardsdatascience.com/weight-initialization-techniques-in-neural-networks-26c649eb3b78`, accessed on 11-11-2019

a kernel frequency estimation. $\hat{p}$ can be seen as the posterior probability that a certain instance should be classed either positive or negative. When $n \leftarrow 0$, PAL is similar to random sampling, since it has no information about the data space (Krempl, Kottke, & Lemaire, 2015). The density, d_x, on unlabeled instances shows the importance of a certain neighborhood. If the neighborhood, thus the instance in this neighborhood, have a high density, it has more influence on a larger part of the dataset and is deemed more important. The probabilistic gain, $pgain(ls)$, is the expected cost reduction of selecting a certain candidate instance. Combining the probabilistic gain and the density weights the optimal candidate is selected for labeling.

The psuedo-code for the PAL algorithm is given in Algorithm 1. Iterating over the data pool $\mathcal{U}$, for each labeling candidate x, the label statistics ls_x, the density weight d_x, and the probabilistic gain $pgain$, are calculated to get g_x (lines 1-6). Finally, the candidate with the highest g_x is selected (line 7).

Algorithm 1 Probabilistic active learning

```
Input:   $\mathcal{U}$   // Unlabeled pool
         $\mathcal{L}$   // Labeled pool
Output:  $x$   // Index of the optimal candidate

1: for $x \in \mathcal{U}$ do
2:     $(n, \hat{p}) \leftarrow labelstatistics(x, \mathcal{L})$
3:     $d_x \leftarrow densityweight(x, \mathcal{L} \cup \mathcal{U})$
4:     $g_x \leftarrow pgain(n, \hat{p}) \cdot d_x$
5: return $\arg\max_{x \in \mathcal{U}}(g_x)$
```

Dimensionality reduction. As stated previous, an image is the same as a MxN matrix of features. Each feature equals one dimension. Working with high dimensional data is computationally very slow or infeasible (i.e. curse of dimensionality) (Verleysen & François, 2005). To address this problem a dimensionality reduction technique called Principal Component Analysis (PCA) can be used (Wold, Esbensen, & Geladi, 1987). PCA is very well suited for dimensionality reduction, but reducing dimension always results in a loss of information. This research uses PCA to calculate the density weights for the PAL method. Figure 9 is used to illustrate what PCA does. When having data that is described in multiple dimensions, it is possible to look at it from a central point of view. To reduce the number of dimensions a straight line is fitted through the central point of view that results in a minimal sum of squared error for b or a maximal sum of squared error for a. Statistics usually opt for the first option. However, PCA uses the second option, since it is easier to compute. This can be done for any dimension. These lines are called Principal Components (PC) and are divined by the slope of the line in comparison to each dimension. The straight orange line is used to describe the first dimension and is called PC1. The second straight green line is used to describe the second dimension and is called PC2. For example an image is used with a width of 2 pixels and a height of 3 pixels, $2x3$ matrix. This results in having 6 dimensions. To reduce the six dimensional data space into a two dimensional data space, PC1 and PC2 can be used. Both, PC1 and PC2 consists of six values, one for each original dimension. PCA is done using Singular Value Decomposition (SVD). Using SVD makes sure that the lengths of each PC is scaled to 1. This means that taking a step of exactly 1 on a PC line will result in an increase or decrease of exactly 1. This results in a scaled distance from the data points. This vector of exactly 1 is also known as the eigenvector. This eigenvector can be used to calculate the eigenvalue using a Pythagorean method (see yellow triangle). The eigenvalues are the sum of squared distances from the center point on the PC line to the central point. For the bottom right point this distance is equal to a. Six data points found by reducing the dimensions are shown in Figure 9. Each data point visualizes exactly one image in a two-dimensional hyperspace. To visualize high dimensional data it is often reduced to two- or three-dimensional data, as was done to illustrate PCA. This is due to the fact that humans are only capable to understand visualization of two- or three-dimensional data. However, to calculate the density weights for the PAL algorithm, the dimensions are reduced to ten. This is done to make sure that not too much information will be lost when reducing the dimensions. Using the reduced dimensions, the PAL algorithm can find the importance of the neighborhoods and thus the density weights.

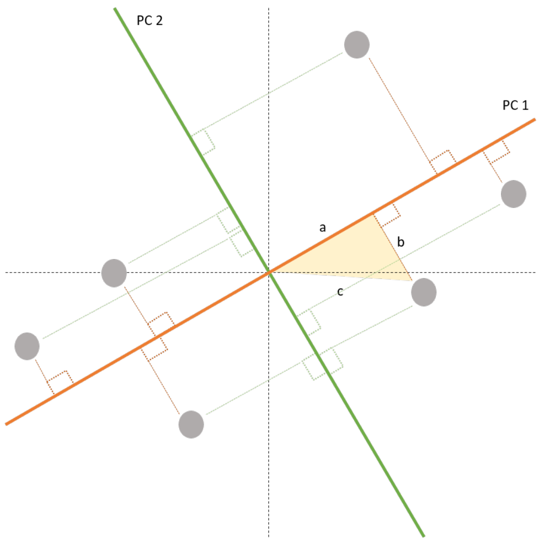


Figure 9: Principal Component Analysis

5. Methodology

This master thesis proposes a tool to use instance segmentation in combination with data augmentation, probabilistic active learning and transfer learning. The data augmentation active and transfer learning instance segmentation (DATALIS) methodology consists of seven steps:

1. Data collection
2. Transfer learning
3. Image annotation
4. Data augmentation
5. Mask RCNN
6. Active learning
7. Prediction

Figure 10 illustrates the methodology applied to building this tool. The visualization of the tool can be found in Appendix A.

In step 1 data is collected. Transfer learning is applied in step 2. In step 3-6 the images are annotated, augmented and run through the Mask RCNN and probabilistic active learning algorithms. This cycle will be repeated until a certain predefined threshold is met. In step 7 a prediction is made on a test dataset to see how well the neural network performs on predicting a class. In the next sections, each step of the methodology will be addressed briefly. However, step 5(Mask RCNN) will be discussed in depth, since this is the most important part of the methodology and has not been covered previously in this master thesis.

5.1. Data collection

Data collection can be done in two ways: (i) using an existing (annotated) dataset, or (ii) collecting new data. Using an existing dataset is useful when testing the performance of a methodology. This is due to the fact that this dataset is not created by the tester and thus is not biased by that particular researcher. Besides, many existing datasets have enough data for training, validating and testing. The second way of obtaining data is by collecting it manually. Collecting data manually is time consuming, but when training a neural network for a specific case, usually manual obtained data is needed. Since this research is focused on images, the collection of images is only addressed. This research focuses on obtaining images through an API. An API is a useful way of interacting with a certain provider to get specific information, in this case images[20]. To obtain data this research uses an API provided by Flickr.com. Through this API, the user can specify how many images should be collected and additional key words can be added. For example when collecting images of a balloon, the user can specify keywords like party or birthday.

In this master thesis two benchmark datasets are used to test how well transfer learning, probabilistic active learning and data augmentation work on instance segmentation. The first dataset contains 61 images of balloons[21] in the training set, 13 in the validation set and 10 in the test set. The dataset has 255, 47 and 136 object instances for the training, val-

[20]https://medium.com/@TebbaVonMathenstien/what-is-an-api-and-why-should-i-use-one-863c3365726b, accessed on 11-11-2019

[21]https://github.com/matterport/Mask_RCNN/tree/master/samples/balloon, accessed on 11-11-2019

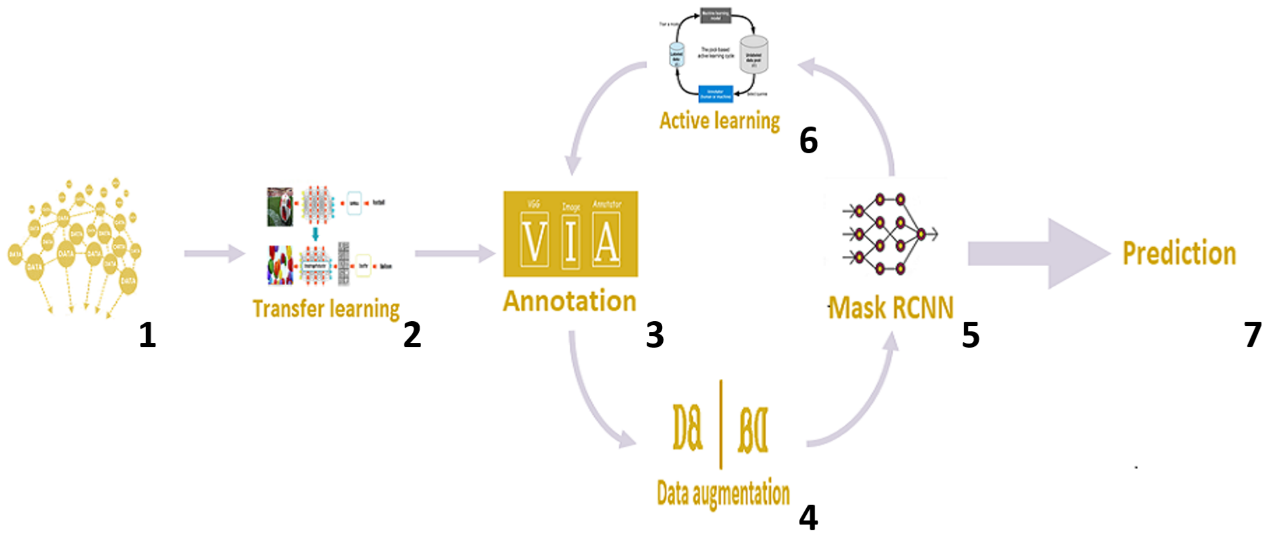


Figure 10: DATALIS methodology.

idation and test set respectively. The second dataset contains 45 images of dogs[22] in the training set, 5 in the validation set and 14 in the test set. Here the training set and validation set only have one object instance on each image, resulting in 45 and 5 object instances respectively. The test set has 19 object instances. Both datasets are very small, since the goal of this research is to see if this tool is useful to use for training neural networks, without much manual effort to label a lot of instances.

5.2. Transfer learning

To test how well transfer learning can be applied on the Mask RCNN model, this master thesis uses weights gathered from a neural network trained on a Common Objects in Context (COCO) dataset (T.-Y. Lin et al., 2014). This COCO dataset can be defined as the source data and was trained on 1.5 million object instances. These objects are visualized in over 200.000 labeled images and consist of 80 different object categories. Transfer learning is well suited when weights are gathered from a source domain that have similar features to the target set (Taylor & Stone, 2009). The balloon target domain is not part of the COCO dataset and thus needs to reshape and use features for other object categories to define the balloons. The dog target domain is already featured in the COCO dataset and thus should be able to use direct features found in the COCO dataset. However, both target domains should be able to use the features of the COCO dataset, since the features found in the source domain are similar to that of the target domain. This master thesis also looks into the usefulness on using weights from a totally different source domain that do not have many similarities. The Mask RCNN model trained on dog weights is used as source domain for a new Mask RCNN model that is used to predict balloons.

[22] https://github.com/RomRoc/maskrcnn_train_tensorflow_colab, accessed on 11-11-2019

In transfer learning there are multiple ways to retrain the gathered weights from the source domain. The two most common ways are by retraining all the weights, or only selecting the end of the neural network (heads) and retraining those (Pan & Yang, 2010). The heads for Mask RCNN are the bounding box regression, classifier and mask heads of the network (see section 5.5). In both cases all the weights are transferred from the source domain to the target domain. However, when only training the heads, all the weights from the earlier stages in the model remain unchanged. When training the whole network, every stage is retrained. This master thesis addresses these two ways to see the difference in performance. These two ways are also compared to transferring the COCO weights, but without retraining the network. This is done to see what the ground prediction of the model is when transferring the weights, without the addition of new training data to reshape the features. The tool gives the user more freedom in choosing how much of the model should be retrained and gives the possibility to retrain any given layer that the user deems useful.

5.3. Data annotation

Data annotation in instance segmenation is the drawing of polygons around the target instances (He, Gkioxari, Dollár, & Girshick, 2017). When using a benchmark dataset (e.g. balloon and dogs dataset), no annotation is needed. These datasets are annotated by the creater of the dataset. However, when training a dataset on a specific domain chosen by the

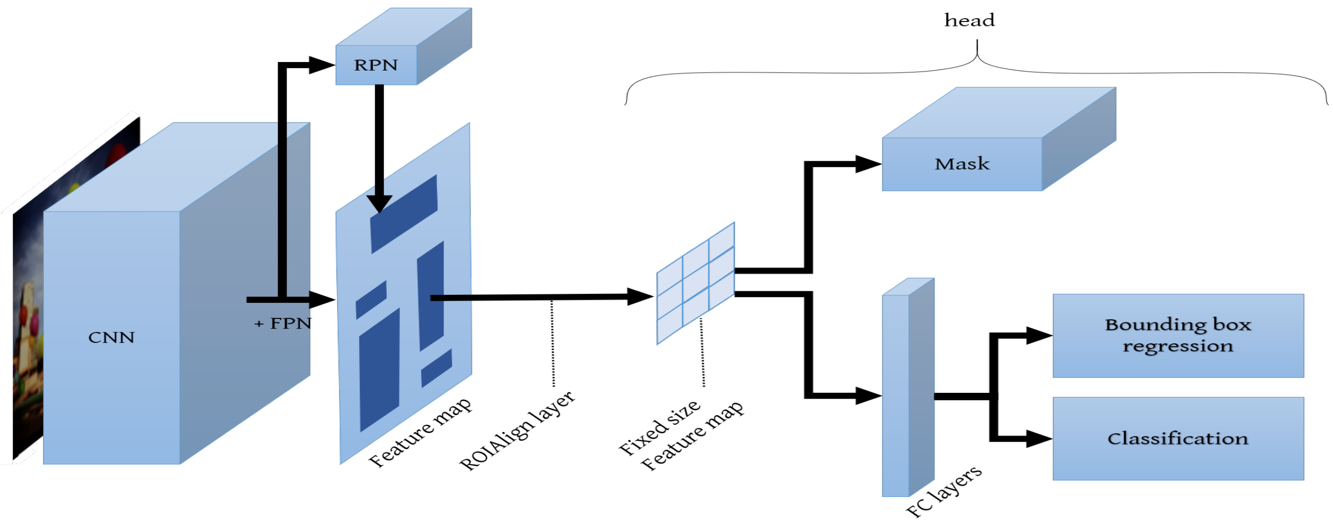


Figure 11: Mask RCNN model

user, images must be annotated. The tool has a build-in annotation application called VGG Image Annotator (VIA) (Dutta & Zisserman, 2019). This tool is easy to use and outputs a json file with the created annotations. The Mask RCNN model can access these json files with ease and uses these to train the model.

5.4. Data augmentation

As described in section 2, nine different data augmentation methods can be used to alter the image. The tool lets the user select the number of augmented images it wants to create for the training set and also gives the option to create augmented images for the validation set. This is useful, since the size of the validation set can improve the performance of a neural network significantly (Guyon, 1997). In this master thesis the performance of the neural network is addressed by looking at augmenting only the training set, and augmenting both the training set and the validation set. For performance measurements, 30 augmentations on each image are performed. This results in 1830 training images and 390 validation images for the balloon dataset. For the dogs dataset, 1350 training images and 150 validation images are created.

5.5. Mask RCNN

The Mask RCNN model is a two stage proposal network. T. Lin, Goyal, Girshick, He, & Dollár (2017) state that two stage detectors outperform more classic object detectors that use a sliding window paradigm. Whereas object detectors that use a sliding window paradigm can find objects rather efficient, two stage proposal networks show for very good accuracy. In the first stage of the Mask RCNN model, images are scanned and areas of interest are generated. In the second stage, objects in these areas are classified through generated bounding boxes and masks. The Mask RCNN model was proposed by He, Gkioxari, Dollár, & Girshick (2017) and extends its predecessor Faster RCNN. The main extension is that Mask RCNN takes pixelwise segmentation into account, instead of only looking at object classification. The Mask MRCNN model is given in Figure 11. In the next sections, the structure of the Mask RCNN is explained through visualization. Finally the configurations used for testing are discussed.

Backbone. The backbone consists of a convolutional nerual network and a Feature Pyramid Network. The convolutional neural network used in Mask RCNN is usually a residual network with 50 or 101 layers (He, Gkioxari, Dollár, & Girshick, 2017). This research uses a residual network with 101 layers, referred to as resnet101 (He, Zhang, Ren, & Sun, 2016). Figure

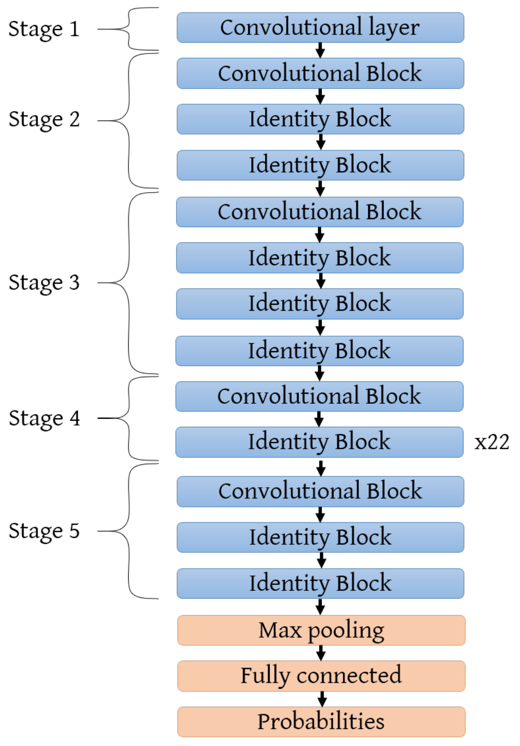


Figure 12: Residual network with 101 layers.

12 shows the structure of the residual network. The network consists of five convolutional stages, followed by a max pooling layer, a fully connected layer and the final predictions.

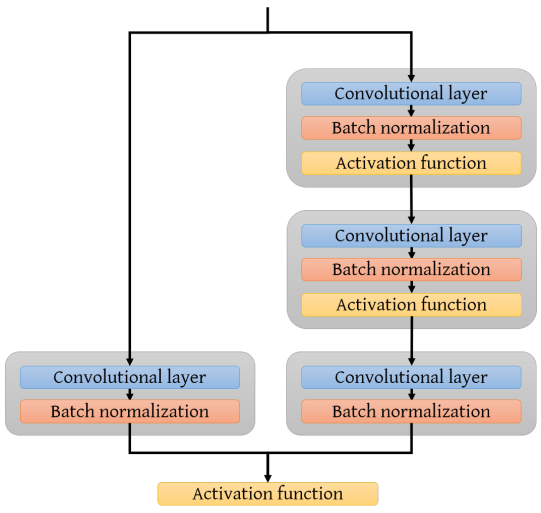


Figure 13: Convolutional block used in a residual network.

Stage 1 takes the images as input, adds zero padding and does 64 $7x7$ convolutions with a stride of 2. Then it adds a batch normalization layer to increase the stability of a neural network and normalizes the output from the previous convolutional layer. Stage 1 ends by applying a ReLU activation function,

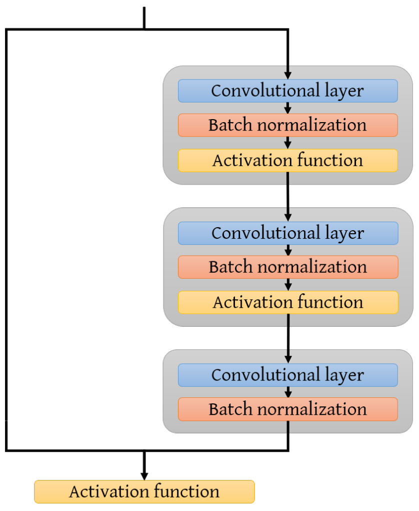


Figure 14: Identity block used in a residual network.

followed by a max pooling layer.

Stage 2 till 5 use convolutional and identity blocks to address the values from the max pooling layer. A convolutional block (Figure 13) is a combination of convolutional layers, batch normalization and activation functions (ReLU). Moreover, it adds a shortcut connection that has a convolutional layer and batch normalization in it. Identity blocks are similar to convolutional blocks. However, the only difference is that the shortcut connections are identity mapping (He, Zhang, Ren, & Sun, 2016), see Figure 14. Stage 2 has one convolutional block, followed by two identity blocks. All three of these blocks uses 256 filters of size 64x64. Stage three has one convolutional block, followed by three identity blocks. Here the blocks use 512 filters of size 128x128. Stage 4 is the biggest stage and has one convolutional block and 22 identity blocks. In this stage 1024 filters of size 256x256 are used for the blocks. The final stage uses 2048 filters of size 512x512 for each block and has one convolutional block and only two identity blocks. The stride is doubled at each stage. Stage 1 started with stride 2, thus stage 2 has a stride of 4. Stage 3-5 have stride 8, 16 and 32 respectively. This means that at each stage the spatial dimension is halved[23].

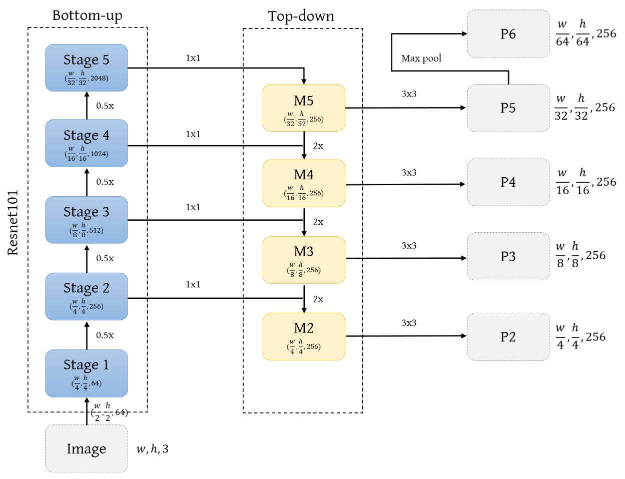

Bottom-up
Top-down
Resnet101
Stage 5 $(\frac{w}{32},\frac{h}{32},2048)$
Stage 4 $(\frac{w}{16},\frac{h}{16},1024)$
Stage 3 $(\frac{w}{8},\frac{h}{8},512)$
Stage 2 $(\frac{w}{4},\frac{h}{4},256)$
Stage 1 $(\frac{w}{4},\frac{h}{4},64)$
$(\frac{w}{2},\frac{h}{2},64)$
Image $w,h,3$
0.5x
1x1
2x
M5 $(\frac{w}{32},\frac{h}{32},256)$
M4 $(\frac{w}{16},\frac{h}{16},256)$
M3 $(\frac{w}{8},\frac{h}{8},256)$
M2 $(\frac{w}{4},\frac{h}{4},256)$
3x3
Max pool
P6 $\frac{w}{64},\frac{h}{64},256$
P5 $\frac{w}{32},\frac{h}{32},256$
P4 $\frac{w}{16},\frac{h}{16},256$
P3 $\frac{w}{8},\frac{h}{8},256$
P2 $\frac{w}{4},\frac{h}{4},256$

Figure 15: Feature Pyramid Network.

The detection of objects at different scales can be challenging when using only convolutional neural networks. To address this problem, Feature Pyramid Networks (FPN) are used (T. Lin et al., 2016). Convolutional neural networks can be seen as a bottom-up approach to find features. FPN uses the output from the convolutional neural network in a top-down pathway[24]. Figure 15 shows this approach. On the left, the five stages from the residual network are shown. The outputs from each stage are used in the FPN to find features. From stage 5, a 1x1 convolution filter is applied to create M5. This is the first feature map for object prediction. Going down the top-down path, the previous layer is upsampled by 2, using nearest neighbors upsampling. Nearest neighbors upsampling is a simple technique and uses pixel duplication to create new higher resolution images (Mazzini, 2018). These higher resolution images are added element-wise to the feature maps obtained at the corresponding stages from the residual network. However, stage 1 is not considered with FPN, since the spatial dimension is too large. Thus, slowing down the process and making it unfeasible. After obtaining all the merged layers, a 3x3 convolution is applied to each layer to obtain the pyramid feature maps, P2-P5. P6 is calculated by using a maximum pooling operation on P5. To illustrate the scale of the input image and how the feature maps size varies during the process, the dimensions are also given in Figure 15. The dimensions are given by the width w, the height h, and the channel c: (w, h, c).

Region proposal network. FPN is used to find features and is not an object detector by itself[25]. Thus, a Region Proposal Network (RPN) is used to scan all the FPN top-bottom pyramid feature maps and proposes regions of interest (ROI) that could contain objects. To bind the objects in the feature maps, anchors are used by the RPN. Anchors are a set of boxes that have predefined locations and sizes relative to the images. Anchor ratios are given to get a set of boxes that have a slightly different width to height ratio. An anchor stride of 1 is used in this master thesis. This is done to be sure that an anchor is created for each cell in the backbone feature map. RPN runs a

[23]`https://medium.com/@fractaldle/mask-r-cnn-unmasked-c029aa2f1296`, accessed on 11-11-2019
[24]`https://medium.com/@jonathan_hui/understanding-feature-pyramid-networks-for-object-detection-fpn-45b227b9106c`, accessed on 11-11-2019
[25]`https://medium.com/@jonathan_hui/understanding-feature-pyramid-networks-for-object-detection-fpn-45b227b9106c`, accessed on 11-11-2019

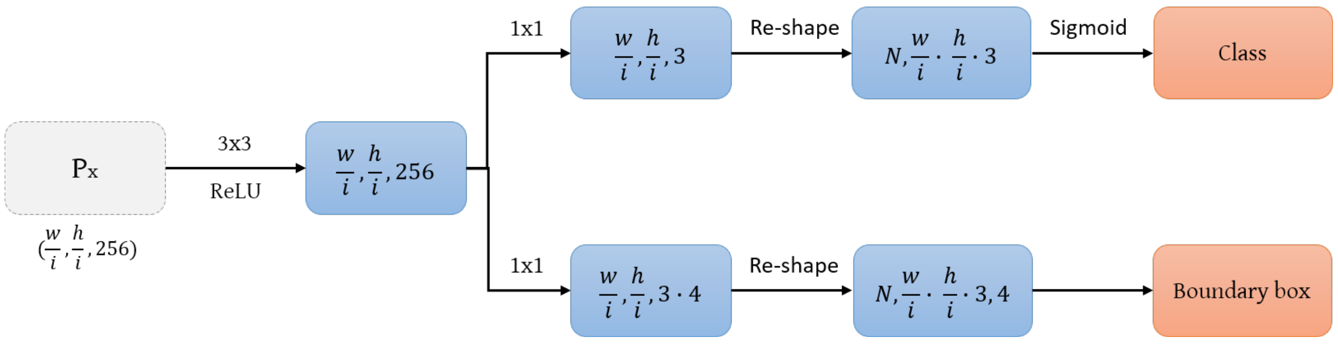

Px
$(\frac{w}{i}, \frac{h}{i}, 256)$
3x3
ReLU
$\frac{w}{i}, \frac{h}{i}, 256$
1x1
$\frac{w}{i}, \frac{h}{i}, 3$
Re-shape
$N, \frac{w}{i} \cdot \frac{h}{i} \cdot 3$
Sigmoid
Class
1x1
$\frac{w}{i}, \frac{h}{i}, 3 \cdot 4$
Re-shape
$N, \frac{w}{i} \cdot \frac{h}{i} \cdot 3, 4$
Boundary box

Figure 16: Region Proposal Network head - Generate a class and boundary box prediction for each pyramid feature map.

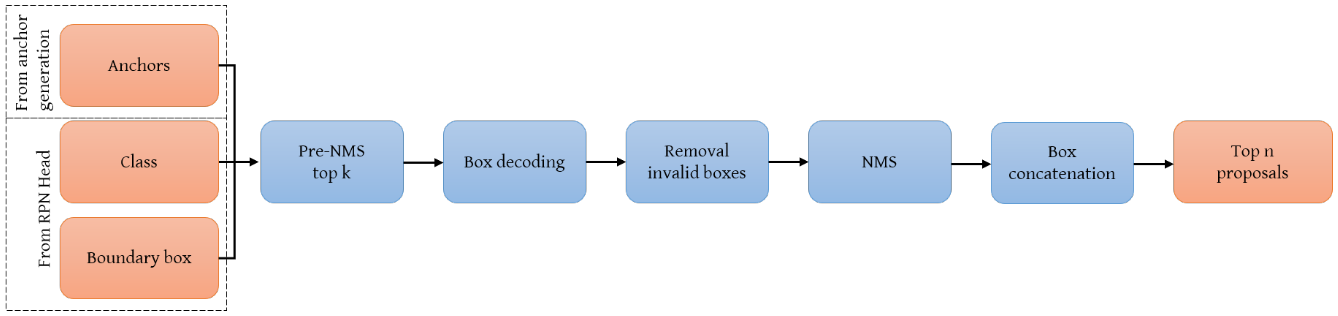


Figure 17: Region Proposal Network - Generate proposals from the anchors, class and boundary box predictions.

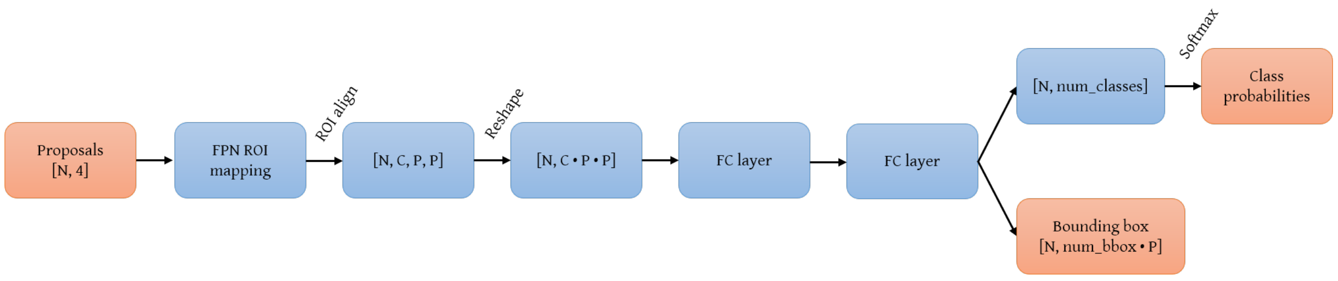


Figure 18: Box head predictions.

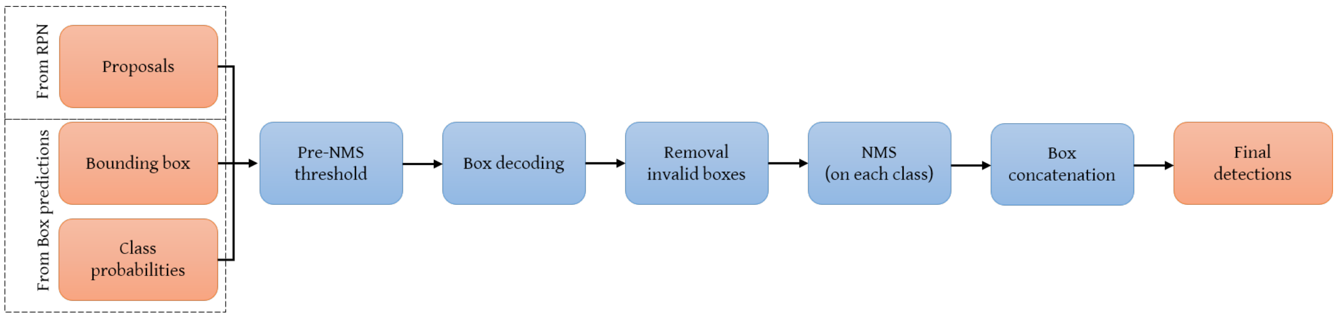


Figure 19: Box final detections.

binary classifier over these anchors in an image[26]. Anchors with a high possibility of containing objects are passed to the second stage of the Mask RCNN model. Remember that the first stage only scans the images and finds areas of interest. The second stage is where objects are classified.

Figure 16 illustrates how the RPN takes the pyramid feature maps from the FPN as input. Each feature map is passed through a $3x3$ convolution layer. The output is passed and transformed to generate a class and bounding box prediction. N is the batch size and the numbers 3 and 4 depict the number of anchor ratios and the coordinates respectively.

The output, together with the generated anchors, are used to generate proposals. Figure 19 illustrates how the generation of proposals is done. First, k anchors are selected that have a good objectness score. The standard value for k is 12000 for training. Secondly, the anchor boxes are modified according to the boundary box output from the RPN head. The third step is to remove invalid boxes. Then non-maximum suppression is applied to remove boxes that are too similar to each other. When boxes have a higher overlap than the threshold $t = 0.7$, a box is removed. Since previous steps are performed separately for each feature pyramid, the fifth step is to concatenate each anchor box. This can be done, since no information of FPN layer origin is needed for the ROI anymore. The final step is to select a top n proposals based on the corresponding objectness score. Default is to set $n = 2000$ for training. This top proposals is selected based on the whole batch.

[26]`https://medium.com/@alittlepain833/simple-understanding-of-mask-rcnn-134b5b330e95`, accessed on 11-11-2019

Regions of interest classifier and Bounding Box regressor. The ROIs obtained by the RPN first have to be mapped to the specific feature maps from the FPN. Only four (P2-P5) are used for this association, since P6 is specifically used to obtain ROIs by the RPN. Equation 4 shows how the feature maps are chosen based on the ROIs width w and height h.

$$k = k_0 + \log_2 \left(\frac{\sqrt{wh}}{224}\right) \tag{4}$$

where $k_0 = 4$ and k is the P_k layer in the FPN. So if $k = 3$ for example, the feature map P3 is chosen.

Since the ROIs are all of a different shape, a uniform shape has to be created. This is done through a ROI Align operation. In earlier models (Fast and Faster RCNN) (Girshick, 2015; Ren, He, Girshick, & Sun, 2015), this was done through a ROI pooling operation. Both the ROI align and the pool operations create a PxP matrix. However, the advantage of using a ROI align operation is that it works well on situations where pixel to pixel correspondence matters. Since this research relies on instance segmentation masks, pixel to pixel correspondence is from utmost importance. After the ROI align operation, multiple fully connected layers and ReLU activation functions are used to output the class probabilities and a bounding box regression in the box head, see Figure 18.

The output from the box head is used, together with the proposals from the RPN, to get the final class and bounding box detections. This is done in a similar way as the RPN (see Figure 17), but sets the non-maximum suppression threshold, $t = 0.5$. Figure 19 illustrates this process.

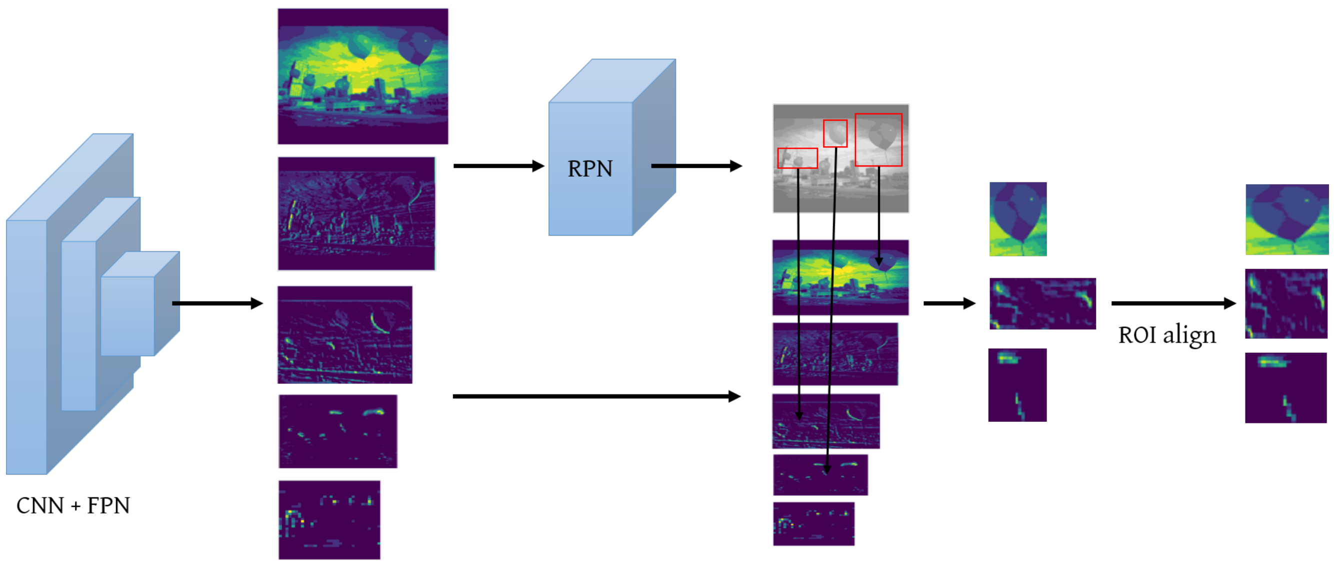


Figure 20: Process on choosing the feature maps based on the output from the FPN.

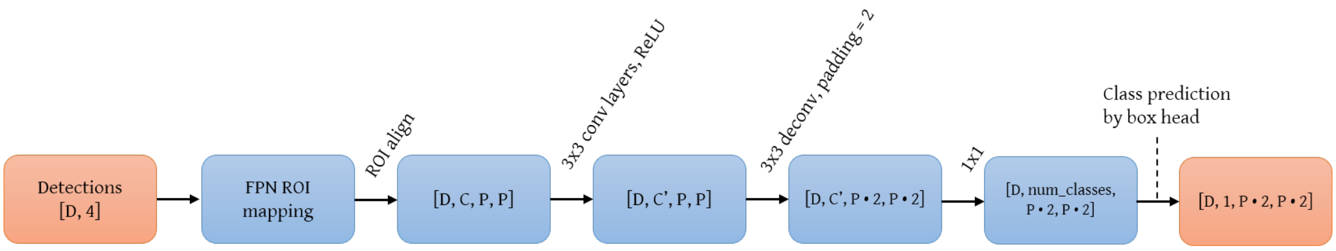


Figure 21: Segmentation mask predictions.

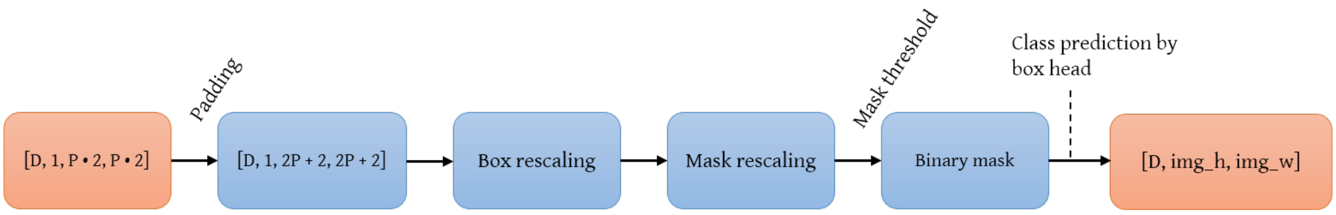


Figure 22: Segmentation mask post-processing.

The process on how the FPN and RPN work together to propose feature maps is visualized in Figure 20[27]. These feature maps are reshaped through the ROI align operation. These final feature maps are used in the object and classification heads.

Segmentation mask. The process of extracting segmentation masks is similar to that of the bounding boxes. However, this process does not make use of fully connected layers, since these lose spatial information. The segmentation mask head starts with a ROI align operation. Instead of passing the output through fully connected layers, multiple convolutional layers are used, followed by a ReLU activation function. For each detection, a deconv operation is added. These features are then passed through a predictor and creates a mask for each class of each detection, see Figure 21. To make sure that the masks are aligned to the coordinates of the image, the masks have to be resized according to that input image. The final step is to apply a binary mask which uses a threshold, $t = 0.5$, to see if a pixel either belongs to the class or to the background. Figure 22 depicts these final steps.

The final three outputs of the model are a bounding box prediction, class probability and a segmentation mask. The jupyter notebook at `https://github.com/jordanvandijk9/Methodology_Mask_RCNN/blob/master/Methodology_visualization.ipynb` visualizes the whole process of the Mask RCNN model and the outputs based on one test image.

Configurations. To train the Mask RCNN model, certain configurations have to be stated. The tool is written in python 3.6.8 with a tensorflow / keras application. To train the neural network an Ubuntu server with two Nvidia Tesla K80's GPUs is used. The GPUs have a size of 12GB and can process two images each at ones. Multiplying the number of GPUs by the image count, gives the batch size. Thus, the batch size that is used for this research, is four. The batch size determines the number of images in one iteration. Each image has a channel count of three (RGB) and optimally has a shape of $1024x1024$, thus a dimension of $[1024, 1024, 3]$.

To train the model, each image is used once for each epoch. Passing the entire training set through the neural network is called an epoch[28]. However, one epoch is not enough for a well trained neural network. This master thesis uses 20 epochs to see how well the neural network performs. When the training dataset is too large, steps per epoch can be defined. This is the number of batch iterations that are performed before an epoch is considered finished[29]. Thus, resulting in only a part of the training data to be used in that epoch. The validation steps is used similarly for the validation set and can also be defined if needed. Since this master thesis uses small datasets, these two configurations are not necessary. A learning rate of 0.0005 is used to make sure that the model converges. Appendix B gives a full overview of all the configurations

[27]`https://medium.com/@jonathan_hui/understanding-feature-pyramid-networks-for-object-detection-fpn-45b227b9106c`, accessed on 11-11-2019

[28]`https://towardsdatascience.com/epoch-vs-iterations-vs-batch-size-4dfb9c7ce9c9`, accessed on 11-11-2019

[29]`https://datascience.stackexchange.com/questions/29719/how-to-set-batch-size-steps-per-epoch-and-validation-steps`, accessed on 11-11-2019

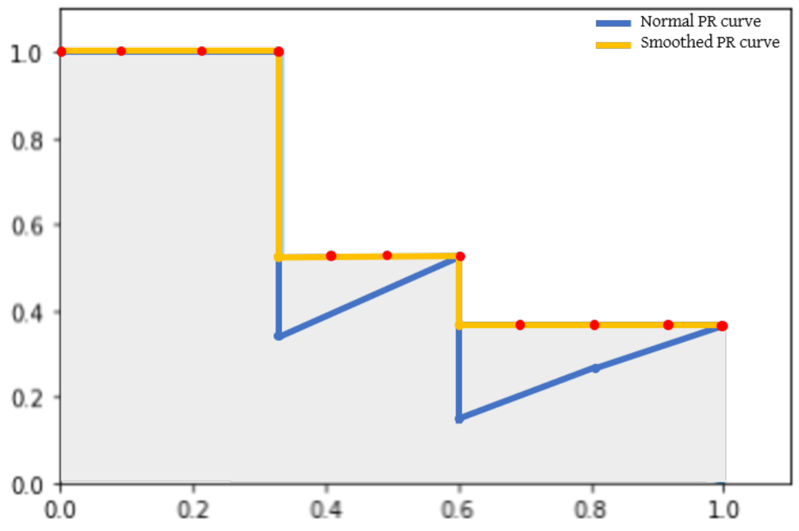


Figure 23: Precision-recall curve shown for, IoU = 0.50 and AP = 0.6273.

5.6. Probabilistic active learning

Probabilistic active learning can be very promising in the field of neural networks when working with a small dataset. The active learner uses the neural network to predict which new image should be labeled. However, neural networks need a lot of data to predict well. These two statements are rather contradicting. To solve this problem, the active learner uses a prediction made by the neural network, only using a small amount of data. By using a prediction, the active learner can choose with a particular certainty the best possible solution. Adding only one new annotated data point to the labeled pool will not increase the performance of the neural network drastically. Therefore, this master thesis uses 30 augmented images of the chosen data point at each new iteration. The active learner is used on a neural network where transfer learning is applied too.

5.7. Prediction

The final step of the methodology is making the prediction. This master thesis assumes that transfer learning will be the most important factor of obtaining good predictions. Neural network with active learning will take long to implement, but might give good results with a very small dataset. Data augmentation will probably reduce overfitting and helps the learner to converge faster. Some limitations of this research can be that training any neural network costs time. Due to the time constraints, running each method multiple times or using epochs is therefore infeasable.

The final prediction is made by using the output of the earlier steps of the methodology. Final predictions use a threshold $t = 0.9$. Any object that is found in an image that has a lower probability than this threshold, is discarded. However, this probability is given by the Mask RCNN model and does not state whether this prediction is correct or not. In instance segmentation, the quality of the prediction is measured by looking at how well the objects are classified and if the pixelwise prediction is similar to the ground truth object. To measure this performance an Average Precision (AP) measurement (T.-Y. Lin et al., 2014) in combination with an Intersection over Union (IoU) (Rezatofighi et al., 2019) is used[30]. The AP uses a precision-recall (PR) curve to calculate the average precision[31]. Figure 23 visualizes the PR curve in blue. To calculate the average precision, the curve is first smoothened. This is done to make sure that the calculated average precision is less suspectable to small variations. The values below the curve are used to calculate the average precision. To obtain the precision and recall (and thus the precision-recall curve), the IoU is used. The IoU measures the overlap between two boundaries, see Appendix C. It measures the overlap of the prediction and the ground truth object. If the overlap is more than a given threshold the prediction is correct. In this master thesis, multiple IoU thresholds are used: $[0.5, 0.6, 0.7, 0.8, 0.9]$. This is done to see if the model is only good at finding the objects or if it also performs well on pixelwise segmentation. Instance segmentation relies on a pixelwise detection of an object. Thus, a neural network that can achieve a high average precision with a high IoU is deemed more useful. However, often only the objects have to be found for deep learning tasks. The precise loca-

[30]`https://medium.com/@jonathan_hui/map-mean-average-precision-for-object-detection-45c121a31173`, accessed on 11-11-2019

[31]`https://medium.com/@timothycarlen/understanding-the-map-evaluation-metric-for-object-detection-a07fe6962cf3`, accessed on 11-11-2019

Table 1: Balloons: AP and IoU

	IoU				
Training	**0.5**	**0.6**	**0.7**	**0.8**	**0.9**
Full neural network	0.0529	0.0274	0.0003	0.0003	0.0000
Only coco	0.0331	0.0181	0.0147	0.0000	0.0000
Transfer learning heads	0.8861	0.8761	0.8558	0.8170	0.4842
Transfer learning all	0.8974	0.8974	0.8974	**0.8691**	**0.6956**
Augmentation	0.0255	0.0176	0.0000	0.0000	0.0000
Augmentation training	0.0697	0.0447	0.0333	0.0000	0.0000
Augmentation TL heads	0.8616	0.8616	0.8616	0.7945	0.4402
Augmentation TL all	**0.9039**	**0.9039**	**0.9018**	0.8517	0.3263
Active learning TL all, Aug$_{(11)}$	0.8658	0.8658	0.8408	0.7692	0.3293
Random learning TL all, Aug$_{(11)}$	0.8431	0.8431	0.8156	0.7678	0.4338

tion is of less importance and thus a lower IoU with a high average precision is good enough. This master thesis makes a distinction between neural networks well suited for object detection [IoU = 0.5, 0.6, 0.7], and neural networks that are best for instance segmentation tasks [IoU = 0.8, 0.9]. This does not mean that a high average precision for one of the categories means that a neural network does not perform well for the other field. Both fields are very closely linked and a high average precision for each IoU is optimal. Besides, a high average precision for a higher IoU is infeasible if the results from lower IoUs are low. Appendix D gives an example of the overlap between the prediction and the ground truth values.

6. Presentation and analysis of the data

In this section explanation and visualiziation of the results will be given. The results are based on the different elements of the methodology. First, the different machine learning methods will be compared. This is done by training the model ten times on each dataset. The results will be compared based on the AP and its values of five different IoU thresholds. Secondly, the effect of probabilistic active learning is discussed in depth. Thirdly, this master thesis looks into the effects of transfer learning. Finally, the training and validation loss(es) will be given and discussed.

Comparison of different machine learning methods. Table 1 and Table 2 show the average precision of different IoU threshold values for multiple neural networks trained on the balloon and the dog dataset respectively. The average precision is calculated after 20 epochs for each neural network. For each dataset, this master thesis looks at the performance of (i) a normal neural network (random initialization of weights), (ii) a neural network that has its weights initialized by the COCO dataset, (iii) a neural network with transfer learning applied only on the heads, (iv) a neural network with transfer learning applied on each layer, (v) a neural network where each image is augmented 30 times for the training and validation set, (vi) a neural network where each image is augmented 30 times for only the training set, (vii) a neural network that uses a combination of transfer learning applied to the heads and data augmentation, (viii) a neural network that uses a combination of transfer learning applied to each layer and data augmentation, (ix) active learning (with transfer learning on all layers and data augmentation), and (x) random learning (with transfer learning on all layers and data augmentation).

Looking at Tables 1 and 2 , it is clear that a neural network without transfer learning (i, ii, v, and vi) does not result in a high average precision for any IoU. Where using only the coco weights (ii) result in the highest value of these bad cases. With a 19.9% average precision found on the dog dataset on both an IoU of 0.5 and 0.6. Whereas only using augmentation for the dog dataset results in a neural network that is not able to correctly identify a single object. For the balloon dataset, using transfer learning on all the layers with augmentation gives the best performance on finding the objects with an IoU of 0.5, 0.6, and 0.7. However, transfer learning on all the layers without augmentation gives the best results when looking at an IoU of 0.8 and 0.9. This can also be said about the dog dataset where augmentation overall resulted in the highest average precision based on the object detection IoU values. Using no data augmentation slightly outperformed augmentation on the instance segmentation task.

Based on the average precision of each IoU, no clear distinction can be made between the methods that use transfer learning. Each method has a similar performance. Using only the heads gives a slightly better result for the dog dataset. However, using the whole

Table 2: Dogs: AP and IoU

Training	IoU 0.5	0.6	0.7	0.8	0.9
Full neural network	0.1571	0.0714	0.0357	0.0000	0.0000
Only coco	0.1990	0.1990	0.1710	0.0752	0.0000
Transfer learning heads	**0.9405**	0.8810	0.6667	**0.5952**	**0.0714**
Transfer learning all	0.8333	0.7976	0.6548	0.5714	**0.0714**
Augmentation	0.0000	0.0000	0.0000	0.0000	0.0000
Augmentation training	0.0000	0.0000	0.0000	0.0000	0.0000
Augmentation TL heads	**0.9405**	0.8095	**0.7381**	0.4048	0.0000
Augmentation TL all	0.8452	0.6667	0.5238	0.4524	**0.0714**
Active learning TL all, Aug$_{(11)}$	0.8810	0.7976	0.5833	0.5119	**0.0714**
Random learning TL all, Aug$_{(11)}$	0.9048	**0.9048**	0.6905	0.5179	0.0000

neural network for transfer learning gives a slightly better result on the balloon dataset. Looking at Table 3 it is clear that running one epoch using transfer learning on the heads (with or without data augmentation) costs the least amount of time. Since every neural network runs for 20 epochs, time can be a big factor in choosing a neural network. There was no significant difference found between the different methods that used transfer learning, thus the fastest method should be chosen. This method is a transfer learning method that uses training only on the heads.

Table 3: Average time spend on running one epoch.

Type	Average time per epoch
Full neural network	640s
Only coco	80s
Transfer learning heads	320s
Transfer learning all	680s
Augmentation	700s
Augmentation training	670s
Augmentation TL heads	350s
Augmentation TL all	710s
Active learning TL all, AUG	980s
Random learning TL all, AUG	840s

Probabilistic active learning vs random learning. Probabilistic active learning (PAL) is proposed as a solution on getting good results with a small amount of data. To see how well it performs, Table 1 and 2 show that PAL has similar results to other methods, but with less images. Whereas the other methods use 61 and 45 images for balloons and dogs respectively. PAL and random learning uses a maximum of eleven images. To see if active selection of new images gives a better performance than just adding random images, this section compares PAL to random learning. Both PAL and random learning involves interaction of the user during the training of the neural network. It either gives the user a new image randomly, or the image is actively selected by the model. Table 3 shows that a user needs to spend around 980 seconds on running one epoch using PAL and 840 seconds on random learning. Table 4 and 5 show the average precision found on the balloon and dog dataset respectively. This is done by looking at the average precision at each iteration. One iteration in this case is applying either active or random learning and running the neural network for two epochs. Both methods start with two annotated images and add one image at each iteration. The annotated image is first augmented 30 times, to make sure that it can have effect on the model. This results in eleven annotated images and 330 augmented images after ten iterations. It can be seen that both PAL and the random learner give good results. Whereas the active learner slightly outperforms the random learner for the balloon dataset, the random learner performs better on the dog dataset. Both dataset perform equally well on object detection. However, it can be stated that training a neural network with interaction from the user performs better on the balloon dataset on instance segmentation tasks. This can be due to the fact that a balloon has a less complex shape than a dog and can be segmented more easily using less data. There was no significant difference between actively selecting a new image to randomly selecting a new image. Besides, active learning takes the longest time to train. To conclude, this master thesis cannot recommend to use PAL for instance segmentation. However, only selecting a few images (actively or randomly) gives similar results to using the whole dataset.

Table 4: Balloons: active learning vs random learning

			IoU		
Epochs	**0.5**	**0.6**	**0.7**	**0.8**	**0.9**
			Active learning		
2	0.8238	0.8158	0.7998	0.7277	0.4470
4	0.8731	0.8694	0.8484	0.8069	0.5330
6	0.8505	0.8505	0.8505	0.8191	0.6418
8	0.9102	0.9102	0.9035	0.8915	0.6328
10	0.9191	0.9191	0.9055	0.8606	0.5296
12	0.8857	0.8834	0.8834	0.8487	0.6184
14	0.8634	0.8634	0.8634	0.8332	0.6138
16	0.8794	0.8794	0.8794	0.8401	0.6014
18	0.9238	0.9188	0.9120	0.8881	0.5681
20	0.8658	0.8658	0.8408	0.7692	0.3293
			Random learning		
2	0.7607	0.7470	0.6975	0.5233	0.1323
4	0.7998	0.7722	0.6683	0.6259	0.2415
6	0.6482	0.5863	0.3887	0.2247	0.1371
8	0.7708	0.7664	0.7257	0.6297	0.4129
10	0.8779	0.8712	0.8443	0.7580	0.3787
12	0.8671	0.8671	0.8483	0.7984	0.5361
14	0.8990	0.8953	0.8827	0.8238	0.5636
16	0.8178	0.8144	0.7952	0.7428	0.4200
18	0.8546	0.8546	0.8521	0.7967	0.3858
20	0.8431	0.8431	0.8156	0.7678	0.4338

Table 5: Dogs: active learning vs random learning

			IoU		
Epochs	**0.5**	**0.6**	**0.7**	**0.8**	**0.9**
			Active learning		
2	0.9048	0.8929	0.5317	0.3889	0.0000
4	0.8373	0.8214	0.5794	0.4544	0.0714
6	0.8810	0.8214	0.5714	0.5000	0.0000
8	0.9405	0.9405	0.8175	0.6508	0.0000
10	0.8611	0.7877	0.7163	0.6329	0.0000
12	0.9603	0.8888	0.7937	0.7143	0.0079
14	0.9246	0.8532	0.7103	0.5516	0.0000
16	0.8333	0.7381	0.6667	0.5556	0.0833
18	0.9286	0.8571	0.7857	0.6429	0.0238
20	0.8810	0.7976	0.5833	0.5119	0.0714
			Random learning		
2	0.9762	0.9405	0.6548	0.3651	0.0000
4	0.9762	0.9762	0.8333	0.6429	0.0000
6	0.9048	0.9048	0.7500	0.5714	0.0119
8	0.8929	0.8214	0.7857	0.6429	0.0714
10	0.8690	0.7619	0.7262	0.7143	0.1429
12	0.9286	0.9286	0.7262	0.5000	0.0794
14	0.9405	0.8690	0.7262	0.5298	0.0000
16	0.9405	0.9405	0.8690	0.7262	0.0238
18	0.9008	0.8294	0.5714	0.4286	0.0238
20	0.9048	0.9048	0.6905	0.5179	0.0000

Table 6: Balloons: transfer learning

	IoU				
Epochs	**0.5**	**0.6**	**0.7**	**0.8**	**0.9**
COCO	0.1990	0.1990	0.1710	0.0752	0.0000
TL H	**0.9405**	0.8810	0.6667	0.5952	0.0714
TL A	0.8333	0.7976	0.6548	0.5714	0.0714
TL DB	0.9037	**0.9037**	**0.8994**	**0.8494**	**0.6997**

Transfer learning. To take a deeper look at how transfer learning performs an, extra overview is given to the main transfer learning results on the balloon dataset. Different to the results from Table 1, here a transfer learning operation is used where the source domain is the dog dataset. The weights from the dog neural network (trained on using transfer learning on all layers of the neural network) are used to see how well it works to transfer learn on balloons. Table 6 shows four different transfer learning results: (i) coco only (COCO), (ii) transfer learning heads from coco weights (TL H), (iii) transfer learning all from coco weights (TL A), and (iv) transfer learning from dogs weights (TL DB). As expected, all three transfer learning methods outperform just using the coco weights to predict the objects. However, surprisingly the TL DB outperforms the other two transfer learning methods on four of the five IoU thresholds. This is surprising since the shapes of dogs and balloons are very dissimilar and Taylor & Stone (2009) states that using transfer learning from similar domains tend to give better performance. A possible explanation can be that the dog weights that are used are obtained after transfer learning from the coco dataset. Since this neural network is also focused on a binary classification problem, it might be easier to fit on a new binary classification problem. There is no significant difference between using transfer learning from the coco or from the dog weights. Both options are feasible and result in faster and better training of a neural network.

Losses. To see what the costs are of training the neural networks, this master thesis uses twelve different losses (He, Gkioxari, Dollár, & Girshick, 2017). The first six losses that are used visualize the validation loss and are given by the (i) overall validation loss, (ii) RPN anchor class validation loss, (iii) RPN bounding box validation loss, (iv) Mask RCNN classification loss, (v) Mask RCNN bounding box refinement loss, and (vi) mask binary cross-entropy loss. The same losses are also visualized for the training loss.

The Mask RCNN classification loss reflects how good the model is at predicting the correct class. Since this master thesis uses a binary classification problem, this loss reflects if the model classifies the balloon or dog correctly. The RPN anchor class loss does something similar, but only looks if it can find an object and not whether this is the particular object that has to be found. This is the reason that the RPN anchor class loss tends to be lower than the Mask RCNN classification loss[32]. The Mask RCNN bounding box refinement loss shows the distance between the predicted bounding box and the ground truth bounding box. It is by nature a regression loss and uses a Smooth L1 loss within and penalizes larger difference exponential. Hence, it shows how good objects can be found within an image. Hereby, the RPN bounding box loss is used to show how well the model finds ROIs in an image. The mask binary cross-entropy loss reflects the pixel-wise classification. It is calculated for each ROI and is only based on masks corresponding to the right class. As stated, these losses are used for both the training and validation loss. The difference in the training and validation loss can be used to see if the model is overfitted.

Looking at Figure 24, it can be seen that the overall validation and training loss tends to be high for (i) a normal neural network, (ii) a neural network where each image is augmented 30 times for the training and validation set, (iii) a neural network where each image is augmented 30 times for only the training set. Looking at the difference between the overall validation loss (10.0) and the overall training loss (7.5), it can be stated that these methods also have a problem of overfitting. This is the reason that the methods do not perform well on unseen data (i.e. test data), see the bad results for average precision in Table 1. The other methods have similar values for the validation losses and the training losses. Interesting to see is that the loss values of each method do not fluctuate much between the epochs. This can be due to wrong hyper parameter settings (e.g. batch size, loss weights, network size). However, the hyper parameters chosen were selected carefully and looking at the values from the loss functions and the average precision the methods have a good performance.

Figure 25 shows the values from the loss functions on training the method on the dog dataset. Similar results can be found for the dog dataset, as the balloon dataset. However, the two methods that use augmentation in combination with either transfer learning implementation, shows signs of overfitting. The overall validation loss is twice as high as the training loss. Surprisingly this does not result in bad average precision scores in Table 2. This is due to the fact that the validation loss mostly differs in the RPN bound-

[32]`https://stackoverflow.com/questions/55360262/what-exactly-are-the-losses-in-matterport-mask-r-cnn`, accessed on 11-11-2019

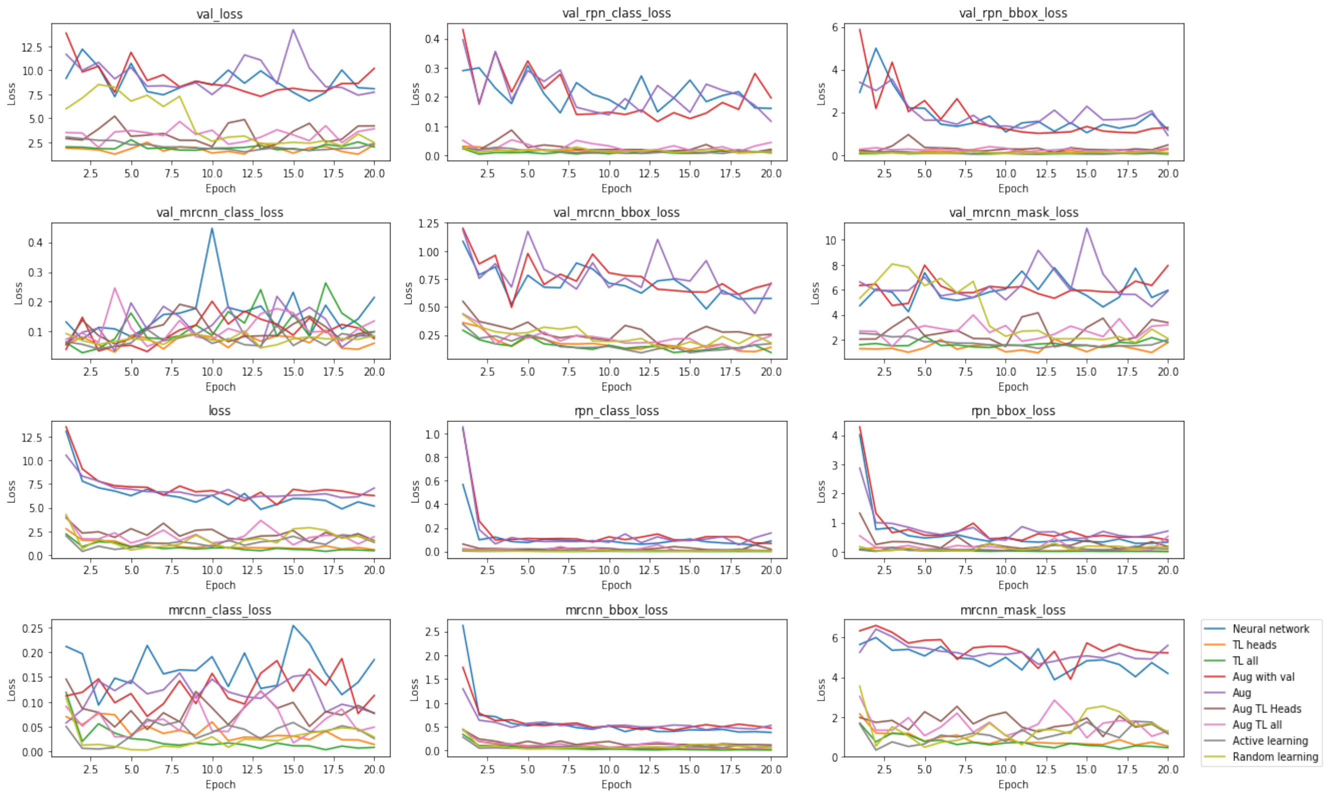


Figure 24: Validation and training loss on the balloon dataset.

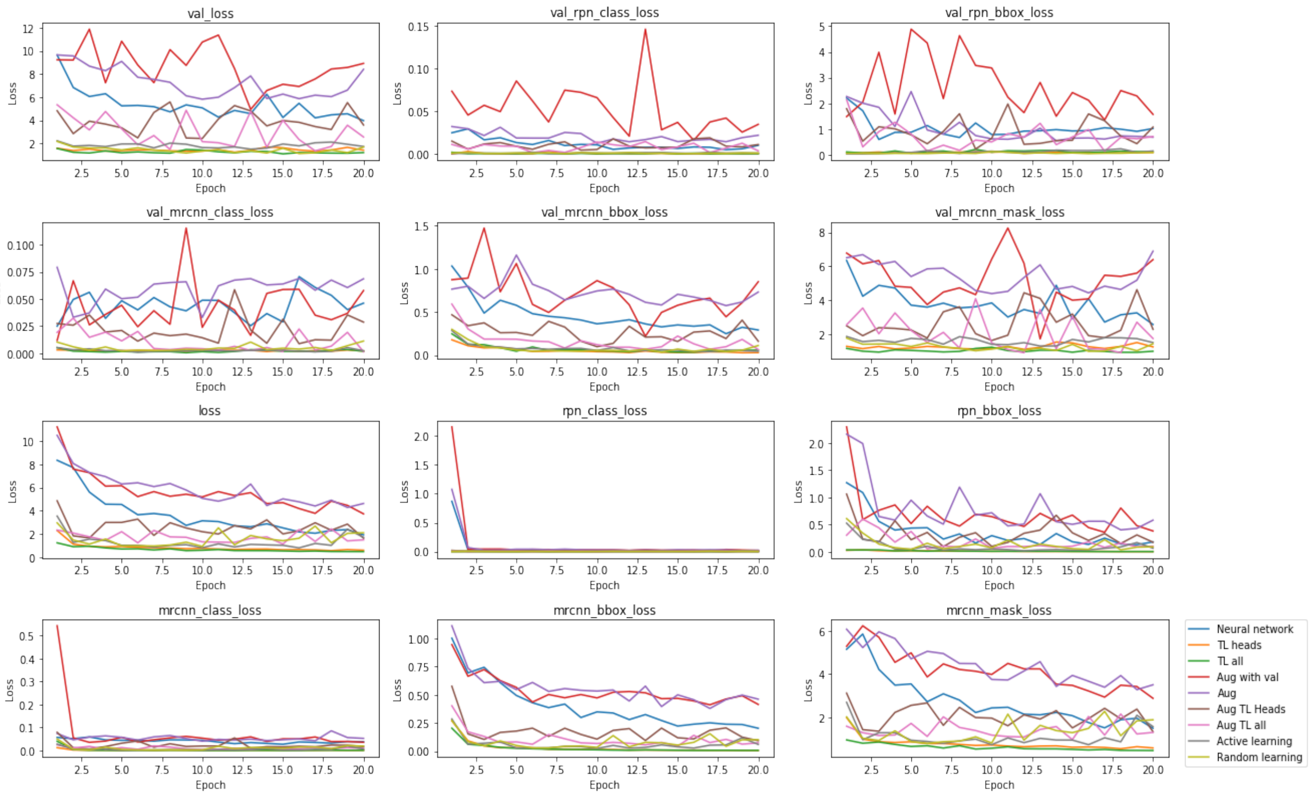


Figure 25: Validation and training loss on the dog dataset.

ing box loss compared to the training loss. This is possibly solved by the MRCNN head when only the dog objects are of importance and not each ROI.

Following the comparison of the different methods based on the average precision and loss function, it can be stated that transfer learning is of utmost importance when having a small dataset. Transfer learning increases the performance and results in low losses, even when only running a few epochs. Data augmentation does not improve the results in this master thesis. PAL also does not improve the results and is not show significant better than random sampling. Interesting is the fact that using only a small subset of images resulted in the same average precision as using the whole dataset. This master thesis recommends to use a transfer learning method only on the heads, with COCO weight initialization, without data augmentation and without PAL.

7. Discussion

The improvement of performance in recent years is mainly due to the existence of larger labeled datasets, more powerful and deeper models, computational excelling machines and better techniques to prevent overfitting. In most real world situations there is a lack of labeled data samples (Otálora, Perdomo, González, & Müller, 2017). A vast amount of data and thus a large amount of variation is also needed to prevent overfitting.

This master thesis addresses the problem of needing a vast dataset to train neural networks for instance segmenation. It also addresses the problem that there does not exist an uniform tool where users can create and use a neural network, without much effort. This master thesis proposes a tool that uses multiple machine learning methods: (i) transfer learning, (ii) data augmentation, and (iii) active learning and is structured in a seven step methodology DATALIS. Furthermore, each of the machine learning methods is tested on the average precision for five Intersection over Union values. Also the loss values are taken into account to see how well a method performs. This master thesis states that transfer learning is the most important factor for getting good results when using a small dataset. This is in accordance to Pan & Yang (2010). This research was expecting to see positive effects on applying data augmentation and active learning. However, the addition of data augmentation or active learning did not change the results significantly.

In future work more tests should be done to see how well the model works on more complex object detection and instance segmentation tasks. The convolutional neural network that is used is already a view years old. Newer versions that use an inception neural network structure are also suitable for instance segmentation and can give slightly better results (Szegedy, Ioffe, Vanhoucke, & Alemi, 2017; Huang et al., 2017).

The probabilistic active learning (Krempl, Kottke, & Spiliopoulou, 2014) has also had a few improvements in the past years and can be suited for multiple classifications (Krempl, Kottke, & Lemaire, 2015; Kottke, Krempl, Lang, Teschner, & Spiliopoulou, 2016). This research focuses on binary classification, but this can be changed in the future. Also the PAL method uses a label statistics, where the label statistics are averaged to calculate one PAL score. However, there is also the option to get a PAL score for each instance and then average these scores. This might result in better suitable images to annotate and results in better performance of the neural network. PAL also makes use of PCA for calculating the density weights. Linderman, Rachh, Hoskins, Steinerberger, & Kluger (2017) states that t-distributed stochastic embedding and auto-encoders (He, Zhang, Ren, & Sun, 2016) can result in better values for dimensionality reduction. This research has only used principal component analysis, which might give non-optimal results. This research can also be applied to videos. This is already been implemented in the code for the tool, but has not been reported and tested. Future work can hopefully prove that data augmentation and probabilistic active learning is useful for the training of convolutional neural networks used for instance segmentation.

References

Ando, R. K., & Zhang, T. (2005). A framework for learning predictive structures from multiple tasks and unlabeled data. *Journal of Machine Learning Research*, *6*(Nov), 1817–1853.

Angluin, D. (1988). Queries and concept learning. *Machine learning*, *2*(4), 319–342.

Atlas, L. E., Cohn, D. A., & Ladner, R. E. (1990). Training connectionist networks with queries and selective sampling. In *Advances in neural information processing systems* (pp. 566–573).

Bengio, Y., Simard, P., Frasconi, P., et al. (1994). Learning long-term dependencies with gradient descent is difficult. *IEEE transactions on neural networks*, *5*(2), 157–166.

Blitzer, J., McDonald, R., & Pereira, F. (2006). Domain adaptation with structural correspondence learning. In *Proceedings of the 2006 conference on empirical methods in natural language processing* (pp. 120–128).

Boser, B. E., Guyon, I. M., & Vapnik, V. N. (1992). A training algorithm for optimal margin classifiers.

In *Proceedings of the fifth annual workshop on computational learning theory* (pp. 144–152).

Caelles, S., Maninis, K.-K., Pont-Tuset, J., Leal-Taixé, L., Cremers, D., & Van Gool, L. (2017). One-shot video object segmentation. In *Proceedings of the ieee conference on computer vision and pattern recognition* (pp. 221–230).

Chapelle, O., Haffner, P., & Vapnik, V. N. (1999). Support vector machines for histogram-based image classification. *IEEE transactions on Neural Networks*, *10*(5), 1055–1064.

Chapelle, O., Scholkopf, B., & Zien, A. (2009). Semi-supervised learning (chapelle, o. et al., eds.; 2006)[book reviews]. *IEEE Transactions on Neural Networks*, *20*(3), 542–542.

Chapelle, O., & Vapnik, V. (2000). Model selection for support vector machines. In *Advances in neural information processing systems* (pp. 230–236).

Ciresan, D. C., Meier, U., & Schmidhuber, J. (2012). Multi-column deep neural networks for image classification. *CoRR*, *abs/1202.2745*. Retrieved from http://arxiv.org/abs/1202.2745

Cohn, D., Atlas, L., & Ladner, R. (1994). Improving generalization with active learning. *Machine learning*, *15*(2), 201–221.

Cohn, D. A., Ghahramani, Z., & Jordan, M. I. (1996). Active learning with statistical models. *CoRR*, *cs.AI/9603104*. Retrieved from http://arxiv.org/abs/cs.AI/9603104

Dai, J., Li, Y., He, K., & Sun, J. (2016). R-FCN: object detection via region-based fully convolutional networks. *CoRR*, *abs/1605.06409*. Retrieved from http://arxiv.org/abs/1605.06409

Dempster, A., Laird, N., & Rubin, D. (1977, 09). Maximum likelihood from incomplete data via em algorithm. *J. Royal Statistical Soc., Series B*, *39*, 1 - 38. doi: 10.1111/j.2517-6161.1977.tb01600.x

Denoeux, T., & Lengellé, R. (1993). Initializing back propagation networks with prototypes. *Neural Networks*, *6*(3), 351–363.

Drago, G. P., & Ridella, S. (1992). Statistically controlled activation weight initialization (scawi). *IEEE Transactions on Neural Networks*, *3*(4), 627–631.

Drucker, H., Burges, C. J., Kaufman, L., Smola, A. J., & Vapnik, V. (1997). Support vector regression machines. In *Advances in neural information processing systems* (pp. 155–161).

Drucker, H., Wu, D., & Vapnik, V. N. (1999). Support vector machines for spam categorization. *IEEE Transactions on Neural networks*, *10*(5), 1048–1054.

Ducoffe, M., & Precioso, F. (2017). Active learning strategy for cnn combining batchwise dropout and query-by-committee. In *Proceedings of the european symposium on artificial neural networks, computational intelligence and machine learning.*

Ducoffe, M., & Precioso, F. (2018). Adversarial active learning for deep networks: a margin based approach. *CoRR*, *abs/1802.09841*. Retrieved from http://arxiv.org/abs/1802.09841

Dutta, A., & Zisserman, A. (2019). The vgg image annotator (via). *arXiv preprint arXiv:1904.10699*.

Fernández-Redondo, M., & Hernández-Espinosa, C. (2001). Weight initialization methods for multilayer feedforward. In *Esann* (pp. 119–124).

Frühwirth-Schnatter, S. (1994). Data augmentation and dynamic linear models. *Journal of time series analysis*, *15*(2), 183–202.

Gal, Y., Islam, R., & Ghahramani, Z. (2017). Deep bayesian active learning with image data. *CoRR*, *abs/1703.02910*. Retrieved from http://arxiv.org/abs/1703.02910

Gatys, L. A., Ecker, A. S., & Bethge, M. (2016, June). Image style transfer using convolutional neural networks. In *2016 ieee conference on computer vision and pattern recognition (cvpr)* (p. 2414-2423). doi: 10.1109/CVPR.2016.265

Girshick, R. B. (2015). Fast R-CNN. *CoRR*, *abs/1504.08083*. Retrieved from http://arxiv.org/abs/1504.08083

Girshick, R. B., Donahue, J., Darrell, T., & Malik, J. (2014). Rich feature hierarchies for accurate object detection and semantic segmentation. *2014 IEEE Conference on Computer Vision and Pattern Recognition*, 580-587.

Glorot, X., & Bengio, Y. (2010). Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics* (pp. 249–256).

Graves, A., Mohamed, A.-r., & Hinton, G. (2013). Speech recognition with deep recurrent neural networks. In *2013 ieee international conference on acoustics, speech and signal processing* (pp. 6645–6649).

Gurney, K. (2014). *An introduction to neural networks.* CRC press.

Guyon, I. (1997). A scaling law for the validation-set training-set size ratio. *AT&T Bell Laboratories*, 1–11.

Hannun, A., Case, C., Casper, J., Catanzaro, B., Diamos, G., Elsen, E., . . . others (2014). Deep speech: Scaling up end-to-end speech recognition. *arXiv preprint arXiv:1412.5567*.

He, K., Gkioxari, G., Dollár, P., & Girshick, R. (2017). Mask r-cnn. In *Proceedings of the ieee international conference on computer vision* (pp. 2961–2969).

He, K., & Sun, J. (2015). Convolutional neural networks at constrained time cost. In *Proceedings of the ieee conference on computer vision and pattern recognition* (pp. 5353–5360).

He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings of the ieee conference on computer vision and pattern recognition* (pp. 770–778).

Holub, A., Perona, P., & Burl, M. C. (2008). Entropy-based active learning for object recognition. In *2008 ieee computer society conference on computer vision and pattern recognition workshops* (pp. 1–8).

Howard, A. G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., . . . Adam, H. (2017). Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*.

Huang, J., Gretton, A., Borgwardt, K., Schölkopf, B., & Smola, A. J. (2007). Correcting sample selection bias by unlabeled data. In *Advances in neural information processing systems* (pp. 601–608).

Huang, J., Rathod, V., Sun, C., Zhu, M., Korattikara, A., Fathi, A., . . . others (2017). Speed/accuracy trade-offs for modern convolutional object detectors. In *Proceedings of the ieee conference on computer vision and pattern recognition* (pp. 7310–7311).

Inoue, H. (2018). Data augmentation by pairing samples for images classification. *CoRR*, *abs/1801.02929*. Retrieved from `http://arxiv.org/abs/1801.02929`

Ioffe, S., & Szegedy, C. (2015). Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*.

Kottke, D., Krempl, G., Lang, D., Teschner, J., & Spiliopoulou, M. (2016). Multi-class probabilistic active learning. In G. A. Kaminka et al. (Eds.), *Ecai* (Vol. 285, p. 586-594). IOS Press. Retrieved from `http://ebooks.iospress.nl/publication/44803`

Krempl, G., Kottke, D., & Lemaire, V. (2015). Optimised probabilistic active learning (opal) for fast, non-myopic, cost-sensitive active classification. *Machine Learning*, 1-28. Retrieved from `http://kmd.cs.ovgu.de/pub/KremplKottkeLemaire2015MACH.pdf` doi: 10.1007/s10994-015-5504-1

Krempl, G., Kottke, D., & Spiliopoulou, M. (2014). Probabilistic active learning: Towards combining versatility, optimality and efficiency. In S. Džeroski, P. Panov, D. Kocev, & L. Todorovski (Eds.), *Discovery science* (pp. 168–179). Cham: Springer International Publishing.

Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems* (pp. 1097–1105).

LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. *nature*, *521*(7553), 436.

LeCun, Y., Bengio, Y., et al. (1995). Convolutional networks for images, speech, and time series. *The handbook of brain theory and neural networks*, *3361*(10), 1995.

LeCun, Y., Boser, B., Denker, J. S., Henderson, D., Howard, R. E., Hubbard, W., & Jackel, L. D. (1989). Backpropagation applied to handwritten zip code recognition. *Neural computation*, *1*(4), 541–551.

Lewis, D. D., & Gale, W. A. (1994). A sequential algorithm for training text classifiers. In *Sigir'94* (pp. 3–12).

Li, J., Luong, M.-T., & Jurafsky, D. (2015). A hierarchical neural autoencoder for paragraphs and documents. *arXiv preprint arXiv:1506.01057*.

Li, X., & Guo, Y. (2013). Adaptive active learning for image classification. In *Proceedings of the ieee conference on computer vision and pattern recognition* (pp. 859–866).

Lin, M., Chen, Q., & Yan, S. (2013). Network in network. *arXiv preprint arXiv:1312.4400*.

Lin, T., Dollár, P., Girshick, R. B., He, K., Hariharan, B., & Belongie, S. J. (2016). Feature pyramid networks for object detection. *CoRR*, *abs/1612.03144*. Retrieved from `http://arxiv.org/abs/1612.03144`

Lin, T., Goyal, P., Girshick, R. B., He, K., & Dollár, P. (2017). Focal loss for dense object detection.

CoRR, *abs/1708.02002*. Retrieved from http://arxiv.org/abs/1708.02002

Lin, T.-Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., … Zitnick, C. L. (2014). Microsoft coco: Common objects in context. In *European conference on computer vision* (pp. 740–755).

Linderman, G. C., Rachh, M., Hoskins, J. G., Steinerberger, S., & Kluger, Y. (2017). Efficient algorithms for t-distributed stochastic neighborhood embedding. *arXiv preprint arXiv:1712.09005*.

Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S. E., Fu, C., & Berg, A. C. (2015). SSD: single shot multibox detector. *CoRR*, *abs/1512.02325*. Retrieved from http://arxiv.org/abs/1512.02325

Long, J., Shelhamer, E., & Darrell, T. (2015, June). Fully convolutional networks for semantic segmentation. In *The ieee conference on computer vision and pattern recognition (cvpr)*.

Martens, J.-P. (1996). A stochastically motivated random initialization of pattern classifying mlps. *Neural processing letters*, *3*(1), 23–29.

Mazzini, D. (2018). Guided upsampling network for real-time semantic segmentation. *arXiv preprint arXiv:1807.07466*.

McCallumzy, A. K., & Nigamy, K. (1998). Employing em and pool-based active learning for text classification. In *Proc. international conference on machine learning (icml)* (pp. 359–367).

Michaelis, C., Ustyuzhaninov, I., Bethge, M., & Ecker, A. S. (2018). One-shot instance segmentation. *arXiv preprint arXiv:1811.11507*.

Otálora, S., Perdomo, O., González, F., & Müller, H. (2017). Training deep convolutional neural networks with active learning for exudate classification in eye fundus images. In *Intravascular imaging and computer assisted stenting, and large-scale annotation of biomedical data and expert label synthesis* (pp. 146–154). Springer.

Pan, S. J., Kwok, J. T., Yang, Q., et al. (2008). Transfer learning via dimensionality reduction. In *Aaai* (Vol. 8, pp. 677–682).

Pan, S. J., & Yang, Q. (2010). A survey on transfer learning. *IEEE Transactions on knowledge and data engineering*, *22*(10), 1345–1359.

Perez, L., & Wang, J. (2017). *The effectiveness of data augmentation in image classification using deep learning.*

Raina, R., Battle, A., Lee, H., Packer, B., & Ng, A. Y. (2007). Self-taught learning: transfer learning from unlabeled data. In *Proceedings of the 24th international conference on machine learning* (pp. 759–766).

Redmon, J., Divvala, S., Girshick, R., & Farhadi, A. (2016). You only look once: Unified, real-time object detection. In *Proceedings of the ieee conference on computer vision and pattern recognition* (pp. 779–788).

Ren, S., He, K., Girshick, R., & Sun, J. (2015). Faster r-cnn: Towards real-time object detection with region proposal networks. In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, & R. Garnett (Eds.), *Advances in neural information processing systems 28* (pp. 91–99). Curran Associates, Inc. Retrieved from http://papers.nips.cc/paper/5638-faster-r-cnn-towards-real-time-object-detection-with-region-proposal-networks.pdf

Rezatofighi, H., Tsoi, N., Gwak, J., Sadeghian, A., Reid, I., & Savarese, S. (2019). Generalized intersection over union: A metric and a loss for bounding box regression. In *Proceedings of the ieee conference on computer vision and pattern recognition* (pp. 658–666).

Romera-Paredes, B., & Torr, P. H. S. (2016). Recurrent instance segmentation. In *Eccv*.

Rumelhart, D. E., Hinton, G. E., & Williams, R. J. (1985). *Learning internal representations by error propagation* (Tech. Rep.). California Univ San Diego La Jolla Inst for Cognitive Science.

Rumelhart, D. E., Hinton, G. E., Williams, R. J., et al. (1988). Learning representations by back-propagating errors. *Cognitive modeling*, *5*(3), 1.

Saxe, A. M., McClelland, J. L., & Ganguli, S. (2013). Exact solutions to the nonlinear dynamics of learning in deep linear neural networks. *arXiv preprint arXiv:1312.6120*.

Schlüter, J., & Grill, T. (2015). Exploring data augmentation for improved singing voice detection with neural networks. In *Ismir* (pp. 121–126).

Schroff, F., Kalenichenko, D., & Philbin, J. (2015). Facenet: A unified embedding for face recognition and clustering. In *Proceedings of the ieee conference on computer vision and pattern recognition* (pp. 815–823).

Sener, O., & Savarese, S. (2017). Active learning for convolutional neural networks: A core-set approach. *arXiv preprint arXiv:1708.00489*.

Settles, B. (2010). *Active learning literature survey* (Tech. Rep.).

Shin, H., Roth, H. R., Gao, M., Lu, L., Xu, Z., Nogues, I., ... Summers, R. M. (2016, May). Deep convolutional neural networks for computer-aided detection: Cnn architectures, dataset characteristics and transfer learning. *IEEE Transactions on Medical Imaging*, *35*(5), 1285-1298. doi: 10.1109/TMI.2016.2528162

Simard, P. Y., LeCun, Y. A., Denker, J. S., & Victorri, B. (1998). Transformation invariance in pattern recognition—tangent distance and tangent propagation. In *Neural networks: tricks of the trade* (pp. 239–274). Springer.

Simonyan, K., & Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*.

Srivastava, R. K., Greff, K., & Schmidhuber, J. (2015). Highway networks. *arXiv preprint arXiv:1505.00387*.

Sugiyama, M., Nakajima, S., Kashima, H., Buenau, P. V., & Kawanabe, M. (2008). Direct importance estimation with model selection and its application to covariate shift adaptation. In *Advances in neural information processing systems* (pp. 1433–1440).

Szegedy, C., Ioffe, S., Vanhoucke, V., & Alemi, A. A. (2017). Inception-v4, inception-resnet and the impact of residual connections on learning. In *Thirty-first aaai conference on artificial intelligence.*

Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., ... Rabinovich, A. (2015). Going deeper with convolutions. In *Proceedings of the ieee conference on computer vision and pattern recognition* (pp. 1–9).

Szegedy, C., Reed, S. E., Erhan, D., & Anguelov, D. (2014). Scalable, high-quality object detection. *CoRR*, *abs/1412.1441*. Retrieved from `http://arxiv.org/abs/1412.1441`

Taigman, Y., Yang, M., Ranzato, M., & Wolf, L. (2014). Deepface: Closing the gap to human-level performance in face verification. In *Proceedings of the ieee conference on computer vision and pattern recognition* (pp. 1701–1708).

Tanner, M. A., & Wong, W. (1987). The calculation of posterior distributions by data augmentation. *Journal of the American Statistical Association*, *82*, 528–540.

Taylor, M. E., & Stone, P. (2009). Transfer learning for reinforcement learning domains: A survey. *Journal of Machine Learning Research*, *10*(Jul), 1633–1685.

van Dyk, D. A., & Meng, X.-L. (2001). The art of data augmentation. *Journal of Computational and Graphical Statistics*, *10*(1), 1-50. Retrieved from `https://doi.org/10.1198/10618600152418584` doi: 10.1198/10618600152418584

Vapnik, V., & Lerner, A. (1963). Generalized portrait method for pattern recognition. *Automation and Remote Control*, *24*(6), 774–780.

Verleysen, M., & François, D. (2005). The curse of dimensionality in data mining and time series prediction. In *International work-conference on artificial neural networks* (pp. 758–770).

Vijayanarasimhan, S., & Grauman, K. (2009). Multi-level active prediction of useful image annotations for recognition. In *Advances in neural information processing systems* (pp. 1705–1712).

Wagner, J., Baur, T., Zhang, Y., Valstar, M. F., Schuller, B., & André, E. (2018). Applying cooperative machine learning to speed up the annotation of social signals in large multi-modal corpora. *arXiv preprint arXiv:1802.02565*.

Wold, S., Esbensen, K., & Geladi, P. (1987). Principal component analysis. *Chemometrics and intelligent laboratory systems*, *2*(1-3), 37–52.

Yam, J. Y., & Chow, T. W. (2000). A weight initialization method for improving training speed in feed-forward neural network. *Neurocomputing*, *30*(1-4), 219–232.

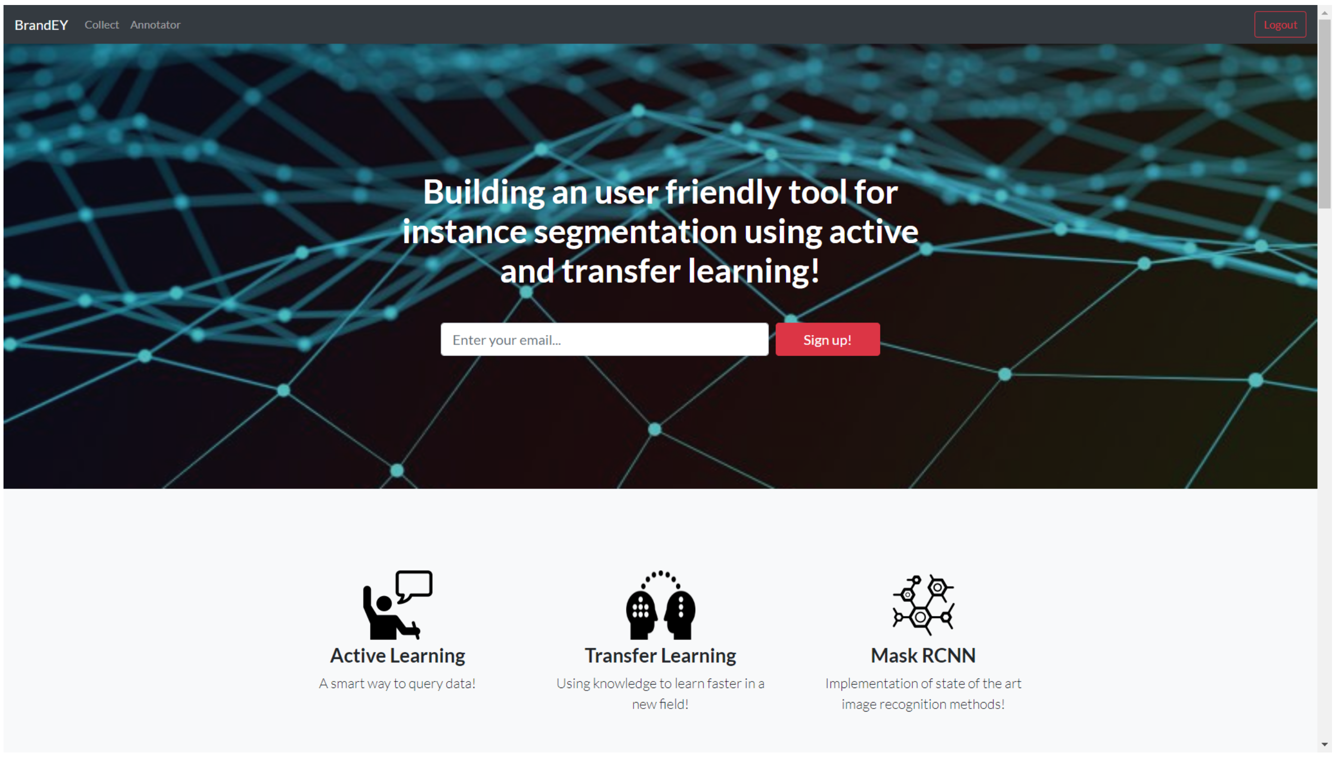
Yosinski, J., Clune, J., Nguyen, A., Fuchs, T., & Lipson, H. (2015). Understanding neural networks through deep visualization. *arXiv preprint arXiv:1506.06579*.

Zeiler, M. D., & Fergus, R. (2014). Visualizing and understanding convolutional networks. In *European conference on computer vision* (pp. 818–833).

Zhong, Z., Zheng, L., Kang, G., Li, S., & Yang, Y. (2017). Random erasing data augmentation. *CoRR*, *abs/1708.04896*. Retrieved from `http://arxiv.org/abs/1708.04896`

Appendices

A. Tool visualization



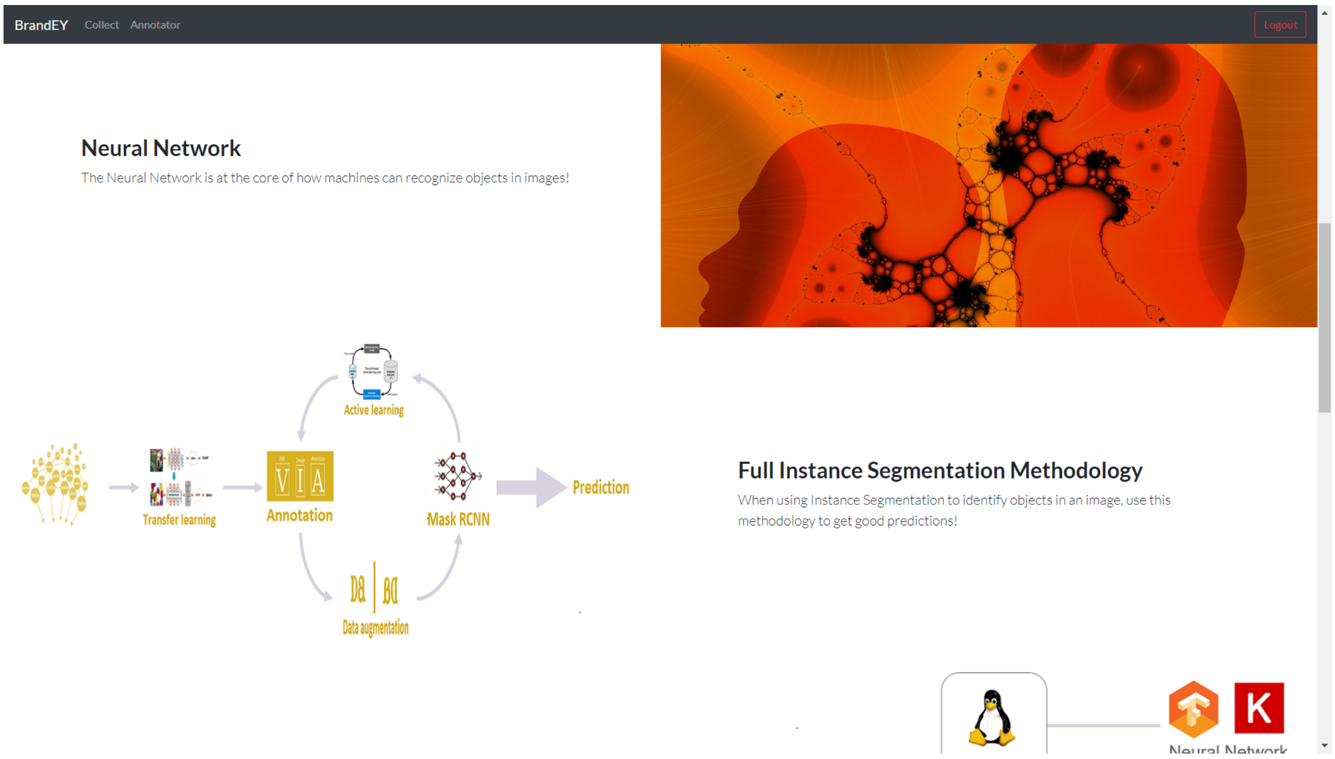
Building an user friendly tool for instance segmentation using active and transfer learning!

A smart way to query data!

Using knowledge to learn faster in a new field!

Implementation of state of the art image recognition methods!

Figure 26: Landing page of the tool (1)



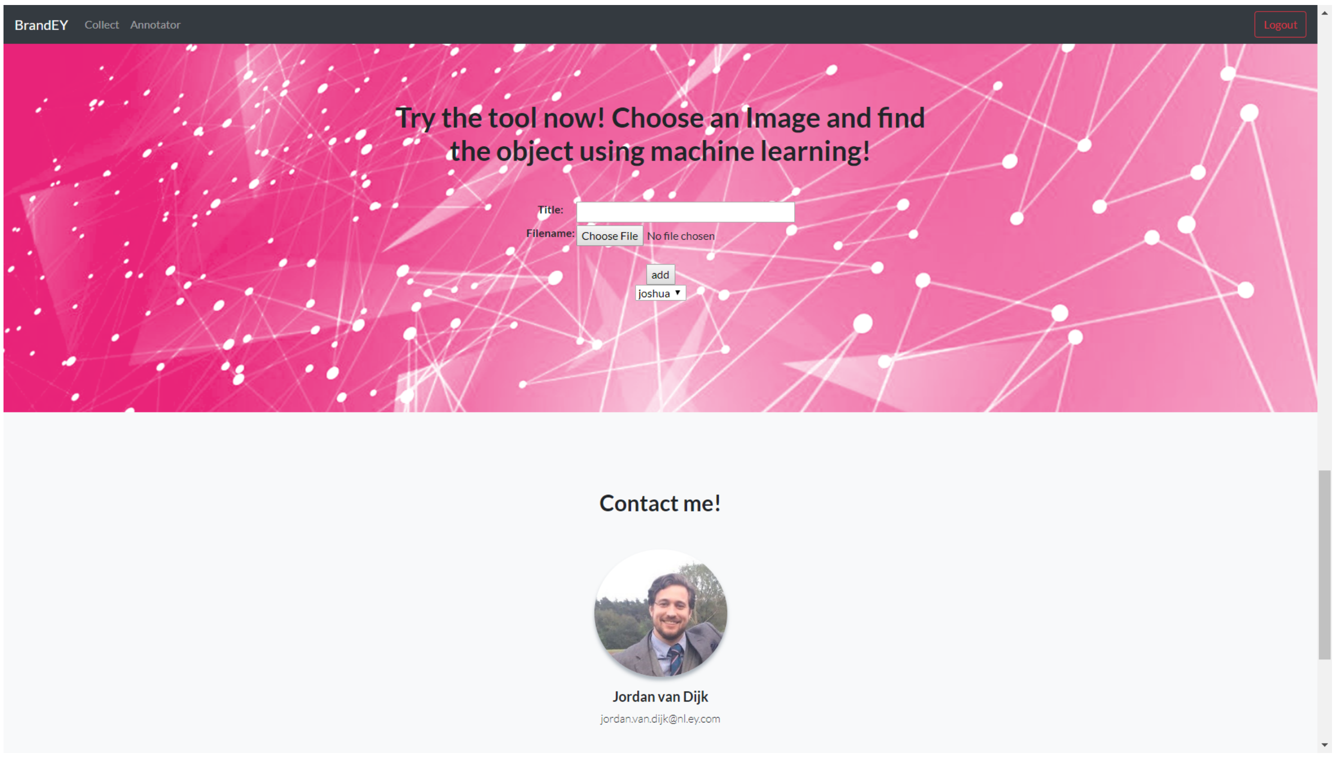
Neural Network

The Neural Network is at the core of how machines can recognize objects in images!

Full Instance Segmentation Methodology

When using Instance Segmentation to identify objects in an image, use this methodology to get good predictions!

Figure 27: Landing page of the tool (2)



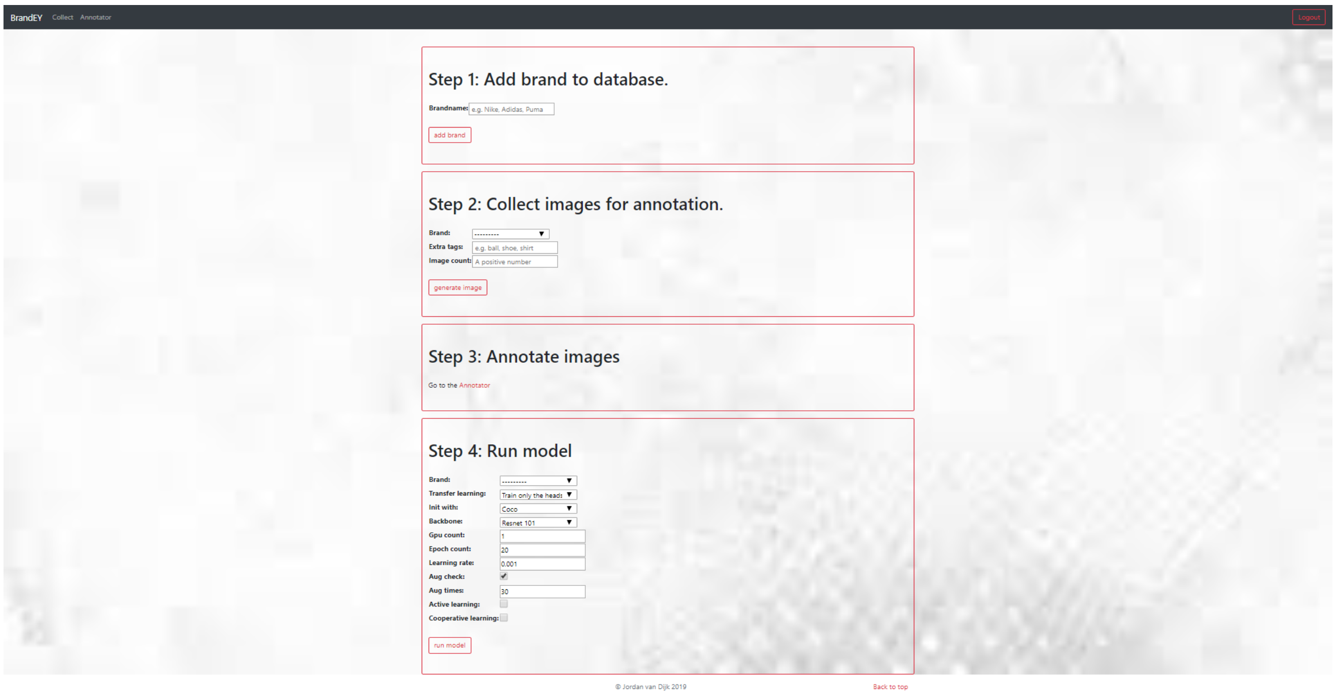
Try the tool now! Choose an Image and find the object using machine learning!

Title:
Filename: No file chosen
joshua

Contact me!

Jordan van Dijk
jordan.van.dijk@nl.ey.com

Figure 28: Landing page of the tool (3)



Step 1: Add brand to database.

Brandname: e.g. Nike, Adidas, Puma

Step 2: Collect images for annotation.

Brand:	---------
Extra tags:	e.g. ball, shoe, shirt
Image count:	A positive number

Step 3: Annotate images

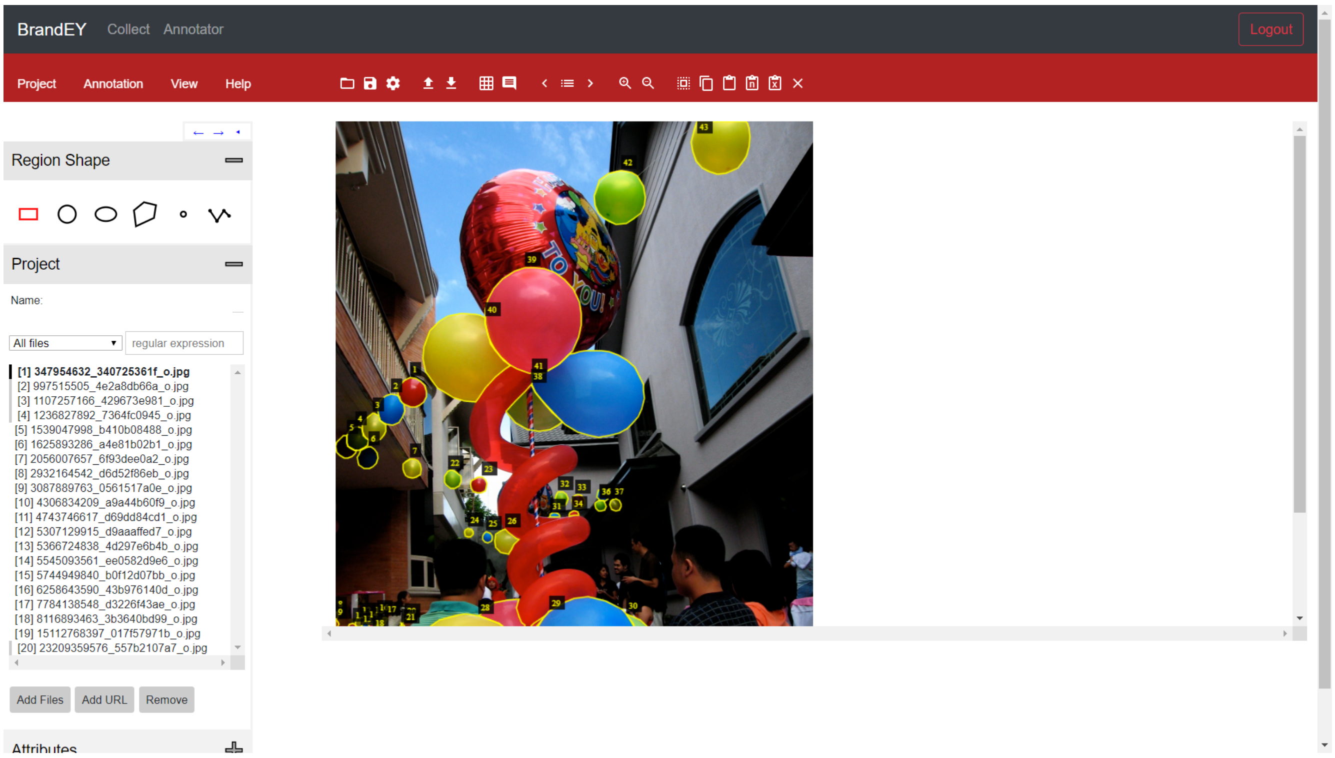
Go to the Annotator

Step 4: Run model

Brand:	---------
Transfer learning:	Train only the heads
Init with:	Coco
Backbone:	Resnet 101
Gpu count:	1
Epoch count:	20
Learning rate:	0.001
Aug check:	✓
Aug times:	30
Active learning:	☐
Cooperative learning:	☐

© Jordan van Dijk 2019

Figure 29: Page that contains the steps to train the Mask RCNN model.



Name:

All files | regular expression

[1] 347954632_340725361f_o.jpg
[2] 997515505_4e2a8db66a_o.jpg
[3] 1107257166_429673e981_o.jpg
[4] 1236827892_7364fc0945_o.jpg
[5] 1539047998_b410b08488_o.jpg
[6] 1625893286_a4e81b02b1_o.jpg
[7] 2056007657_6f93dee0a2_o.jpg
[8] 2932164542_d6d52f86eb_o.jpg
[9] 3087889763_0561517a0e_o.jpg
[10] 4306834209_a9a44b60f9_o.jpg
[11] 4743746617_d69dd84cd1_o.jpg
[12] 5307129915_d9aaaffed7_o.jpg
[13] 5366724838_4d297e6b4b_o.jpg
[14] 5545093561_ee0582d9e6_o.jpg
[15] 5744949840_b0f12d07bb_o.jpg
[16] 6258643590_43b976140d_o.jpg
[17] 7784138548_d3226f43ae_o.jpg
[18] 8116893463_3b3640bd99_o.jpg
[19] 15112768397_017f57971b_o.jpg
[20] 23209359576_557b2107a7_o.jpg

Figure 30: Page where new data can be annotated for the training, validation or test set.

B. Configurations used to train the Mask RCNN model

Table 7: Configurations Mask RCNN

Configurations	**values**
GPU	Nvidia Tesla K80
GPU size	12GB
GPU count	2
Images per GPU	2
Batch size	4
Image channel count	3
Image shape	[1024, 1024, 3]
Image resize mode	Square
Image minimal dimension	800
Image maximal dimension	1024
Backbone	Residual network 101
Backbone strides	[4, 8, 16, 32, 64]
Fully connected layer size	1024
Top down pyramid size	256
Classes (including background)	2
Bounding box standard deviation	[0.1, 0.1, 0.2, 0.2]
Length of square anchor (RPN) stride	[32, 64, 128, 256, 512]
Ratio anchor (RPN) at each cell	[0.5, 1, 2]
RPN anchors per image	256
RPN anchor stride	1
RPN nms threshold	0.7
RPN bounding box standard deviation	[0.1, 0.1, 0.2, 0.2]
ROI positive ratio	0.33
Pool size	7
ROIs per image	200
Pre nms training	6000
Post nms ROI inference	0.33
Post nms ROI training	2000
Mask pool size	14
Mask shape	[28, 28]
Detection minimal confidence	0.9
Detection nms threshold	0.3
Batch normalization	Freeze
Learning rate	0.0005
learning momentum	0.9
Weight decay regularization	0.0001
Gradient norm clipping	5.0
RPN class loss	1.0
RPN bounding box loss	1.0
MRCNN class loss	1.0
MRCNN bounding box loss	1.0
MRCNN mask loss	10.0

C. Intersect over Union visualization

The predictions (yellow and green) are compared to the ground truth object (blue). The green circle has more than 90% overlap with the blue circle and thus will give positive result on every IoU. The yellow circle has more than 50% overlap with the blue circle. However, it does not have more than 60% overlap, thus it will only give positive results on an IoU of 0.5.

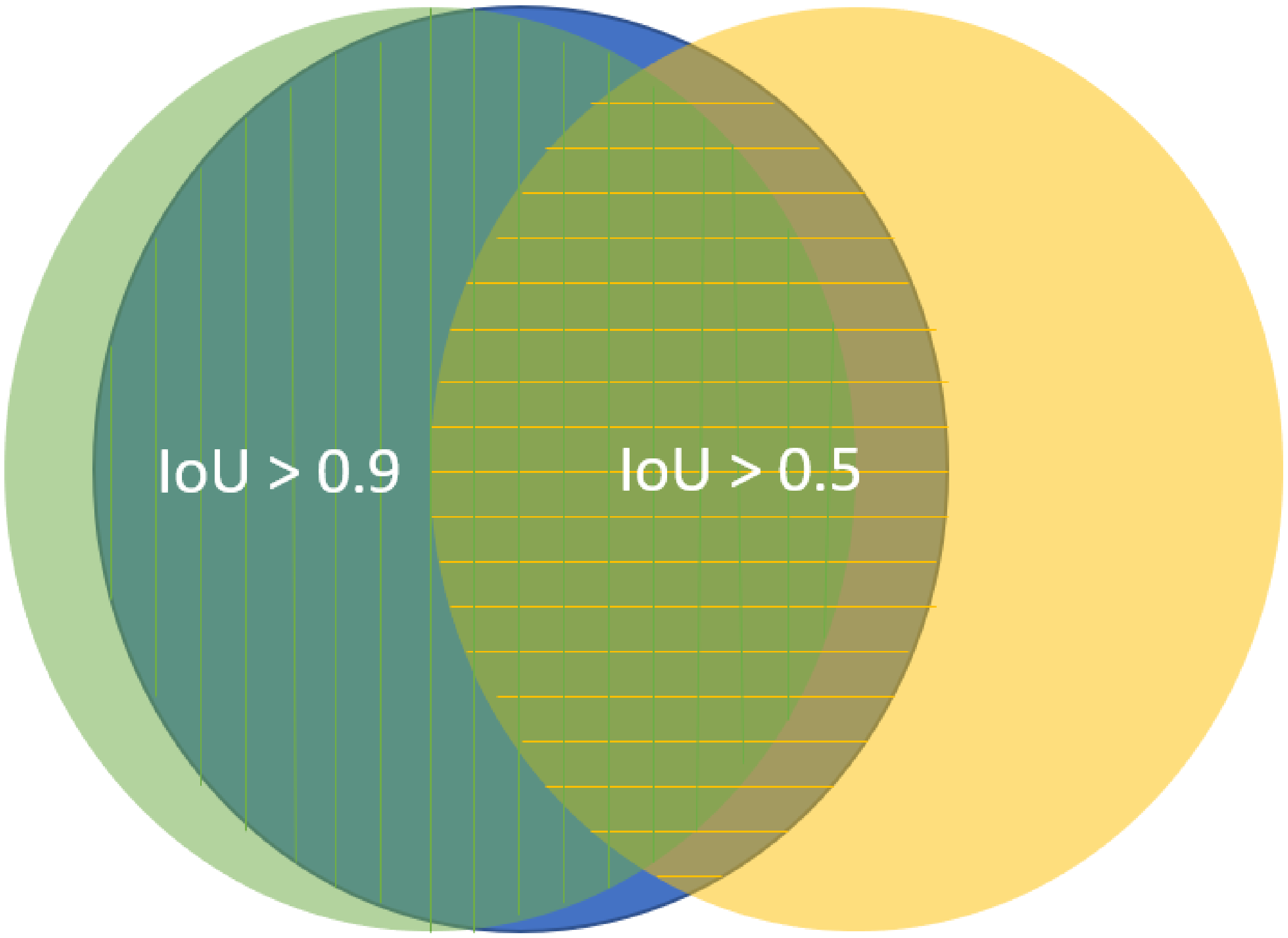


Figure 31: Intersect over Union for a threshold of 0.5 and 0.9.

D. Overlap of predictions and ground truth

The prediction is compared to the ground truth for a certain average precision

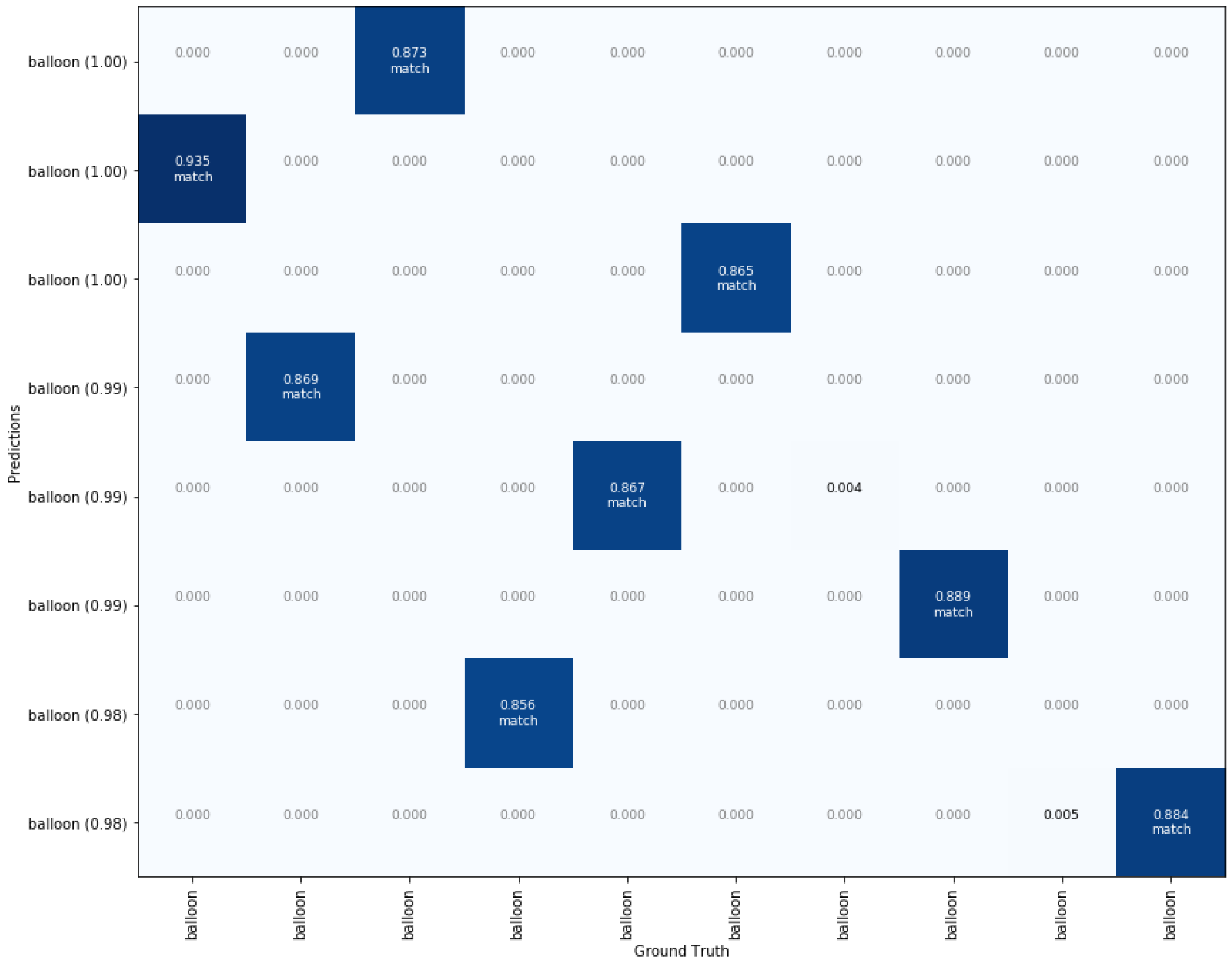


Figure 32: Overlap of the prediction and ground truth.