2018•2019
Faculteit Industriële ingenieurswetenschappen
master in de industriële wetenschappen: elektronica-ICT

Masterthesis

Weakly-Supervised Semantic Segmentation by Learning Label Uncertainty

PROMOTOR :
Prof. dr. ir. Bart VANRUMSTE

PROMOTOR :
dr. ir. Marc PROESMANS

BEGELEIDER :
dr. ir. Bert DE BRABANDERE

Robby Neven

Scriptie ingediend tot het behalen van de graad van master in de industriële wetenschappen: elektronica-ICT

Gezamenlijke opleiding UHasselt en KU Leuven

2018 • 2019

Faculteit Industriële ingenieurswetenschappen

master in de industriële wetenschappen: elektronica-ICT

Masterthesis

Weakly-Supervised Semantic Segmentation by Learning Label Uncertainty

PROMOTOR :

Prof. dr. ir. Bart VANRUMSTE

PROMOTOR :

dr. ir. Marc PROESMANS

BEGELEIDER :

dr. ir. Bert DE BRABANDERE

Robby Neven

Scriptie ingediend tot het behalen van de graad van master in de industriële wetenschappen: elektronica-ICT

UHASSELT KU LEUVEN

Preface

This thesis marks the end of my master's program in Engineering Technology and was conducted from September 2018 until June 2019. Over the last several years, I developed an interest into the popular field of machine learning. To be able to gain a deeper theoretical understanding and get a hands-on experience, I wanted my thesis subject to be within this exciting field. Therefore, I find myself very lucky that I was able to get a thesis proposal from the research group PSI at KU Leuven, which primarily focuses on computer vision using deep learning. The task of finding a weakly-supervised method to train a deep learning network for semantic segmentation was very interesting and enabled me to learn valuable concepts about uncertainty within artificial intelligence.

Of course, I would not have been able to finish this work without the help of many people whom I would like to thank. First of all, I would like to thank my supervisor Marc Proesmans, daily supervisor Bert De Brabandere and internal supervisor Bart Vanrumste for their support and enthusiasm throughout the year. Secondly, I would like to thank my brother. He inspired me to make the switch from Economics to Electrical Engineering and has been my personal mentor throughout the years. Lastly, I would like to thank my friends and family for their emotional, physical and financial support.

Contents

List of Tables

List of Figures

Abstract

Since the rise of deep learning, semantic segmentation tasks have been able to achieve outstanding results. However, the current deep learning approaches need a lot of highly-detailed annotated training data, which are very costly to produce. The aim of this thesis is to develop a new, weakly-supervised method to train a semantic segmentation network using coarse bounding box labels, which are cheaper and faster to obtain. To this end, new loss functions are developed and implemented. The loss functions combine aspects from aleatory uncertainty and online bootstrapping to enable the network to ignore the wrongly-annotated pixels within the bounding box labels during training. First, these principles are applied to the standard regression loss function. Next, since a classification loss function is better suited for segmentation tasks, the same concepts are applied to the binary classification loss function. Both loss functions are used to train a car segmentation network and are evaluated on the Cityscapes dataset. In the last stage of this thesis, the modified binary classification loss is adapted for multi-class segmentation. The new loss is applied to the car/pedestrian segmentation task and evaluated on the Cityscapes dataset. With 18% of the dataset perfectly annotated and the other 82% only containing bounding box annotations, the new loss function performs nearly as good (77.5 vs. 79.8 mIoU) as the standard cross-entropy loss function trained with 100% of the dataset annotated perfectly.

Abstract

Sinds de opkomst van *deep learning* kent het domein van computervisie een enorme vooruitgang op het gebied van semantische segmentatie. Echter, om een hoge performantie te bekomen is een nauwkeurig gelabelde dataset nodig, welke een grote kost met zich meebrengt. Het doel van deze thesis is daarom de ontwikkeling van nieuwe methodes om een deep learning netwerk te trainen aan de hand van *bounding box* annotaties. Dit aangezien dergelijke annotaties sneller en goedkoper te verkrijgen zijn. Hiervoor worden er nieuwe kostfuncties ontwikkeld die via een combinatie van aleatorische onzekerheid en *online bootstrapping* in staat zijn om de slecht gelabelde pixels binnen de *bounding box* annotaties te negeren. In een eerste fase van de thesis worden deze technieken toegepast op de standaard regressie kostfunctie. Vervolgens, omdat een classificatie kostfunctie beter geschikt is voor segmentatie, worden dezelfde concepten toegepast op de binaire classificatie kostfunctie. Beide methoden worden gebruikt om een binair auto segmentatienetwerk te trainen en worden geëvalueerd op de Cityscapes dataset. In de laatste fase van deze thesis wordt de nieuwe binaire classificatie kostfunctie uitgebreid naar een *multi-class* classificatie kostfunctie. Deze wordt vervolgens gebruikt om een auto/persoon segmentatienetwerk te trainen en wordt ook geëvalueerd op de Cityscapes dataset. Met slechts 18% nauwkeurig gelabelde data en 82% *bounding box* labels wordt met de nieuwe kostfunctie een score behaald die vergelijkbaar (77.5 versus 79.8 mIoU) is met een standaard netwerk dat getraind is op 100% nauwkeurig gelabelde data.

Chapter 1

Introduction

1.1 Semantic segmentation

Whereas humans can analyse and understand the context within an image in a single glance, it is one of the most difficult tasks for a machine. Therefore, understanding the context of an image is the main objective of computer vision. In the past, numerous attempts are made to make machines "smarter", by developing algorithms that can analyse images. However, over the last decade, the rise of deep learning has enabled the field of computer vision to excel.

One of the main objectives of computer vision is segmentation. Segmentation is the task of assigning a specific class to each pixel in an image. There are two different types of segmentation: semantic segmentation and instance segmentation. Semantic segmentation is the objective to label individual pixels to a specific class. Instance segmentation goes a bit further, and tries to group pixels from a certain class into different objects. Figure 1.1 gives a clear view of the difference between the two tasks.

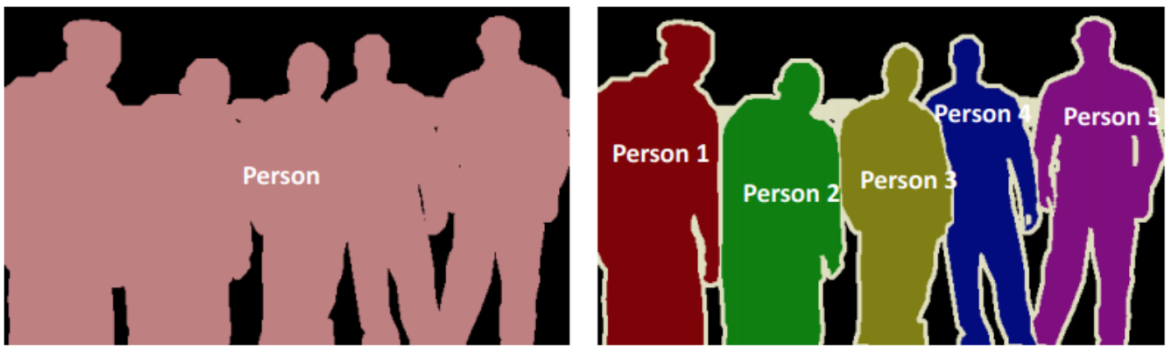


Figure 1.1: This figure is an example of the Pascal VOC dataset [2]. The image to the left shows the semantic segmentation of the image. All pixels belonging to the person class are labelled person. In the image to the right, the person pixels are furthered grouped to individual persons: the different instances. This task is referred to as instance segmentation.

1.2 Problem statement

As mentioned, deep learning enabled computer vision algorithms to achieve outstanding results. However, the standard deep learning approaches are fully supervised. This means

Figure 1.2: Example of a highly-detailed image label from the Cityscapes dataset [1].

they make use of a large labelled dataset to train the deep neural network. To construct such a dataset, one has to label each individual pixel to a specific class. Figure 1.2 shows an example of such a label. Since most of the use-cases for segmentation tasks require images of high resolution (autonomous vehicles, medical imagery, ...), it is clear that this is a time consuming job, making it one of the largest costs in deep learning. For example, one of the most popular image labelling services is Amazon SageMaker Ground Truth. At the time of writing, the cost for labelling an image for semantic segmentation (pixel-perfect label) is $0.840 per labelled segment. Since a standard image on average consists out of 10 segments, the price for a single image is in the range of $8.40. Having in mind the size of an average dataset is a couple of ten thousand images, the total price to label a dataset can be very expensive.

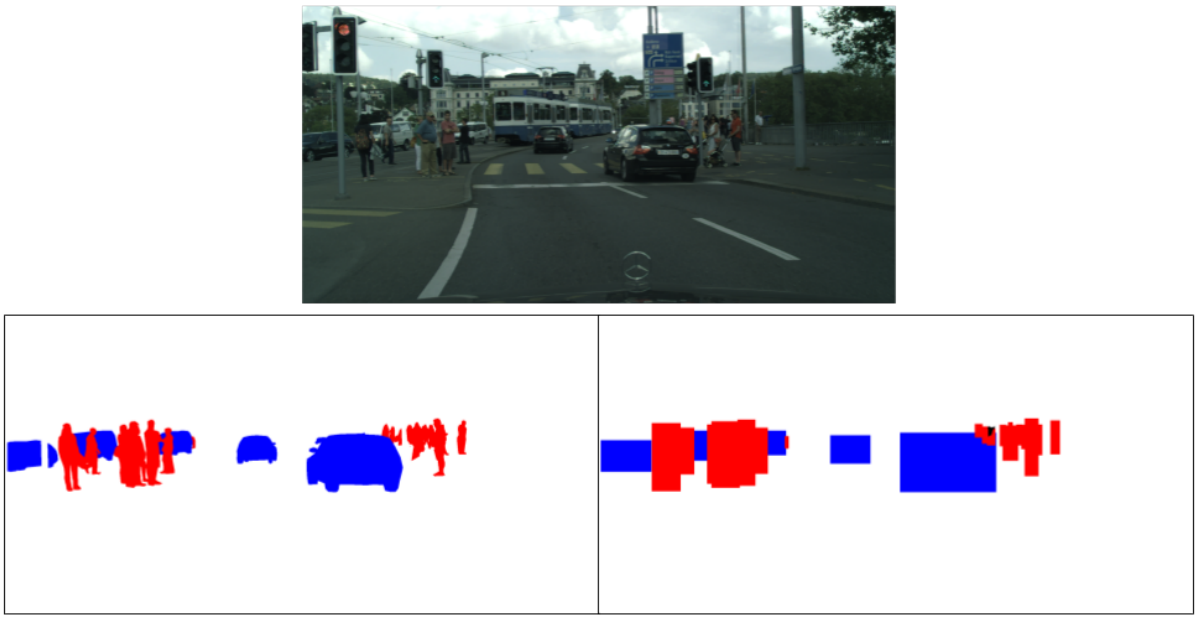
1.3 Objective

Another way of labelling objects is by using bounding boxes: each object is enclosed by a rectangle as seen in figure 1.3. This way of labelling is extremely fast. Whereas for pixel-perfect labels a person has to label each individual pixel to a specific class, they now only have to mark two points of an enclosing rectangle and select the according class. The speed of the labelling is reflected in the price per bounding box: $0.036 per object at Amazon SageMaker Ground Truth, in contrast to $0.840 for a pixel-perfect label. Of course, bounding box labels are not suitable to achieve good segmentation results right out of the box. Since the labels now consist out of a lot of erroneous labelled pixels, training a network with bounding box labels might give unexpected results. The objective of this master's thesis is to train a deep neural network for semantic segmentation using bounding box labels, while still achieving segmentation accuracy as close to a network being trained with pixel-perfect labels.

1.4 Method

In the first stage of this thesis a standard segmentation network will be trained on only bounding box labels. The goal of this stage is to evaluate the segmentation accuracy of a standard training routine in conjunction with the inferior bounding box labels.

In the next stages, we will introduce aleatory uncertainty into the network. The



(a) pixel-perfect label (b) Bounding box label

Figure 1.3: The figure displays the pixel-perfect label as well as the bounding box label for an image.

uncertainty will learn to account for the erroneous pixels from the bounding box labels during training. This will result in some sort of label uncertainty. Having a measure for the label uncertainty, the loss function of the network can take this into account and ignore the uncertain labels within the bounding box label for further training. First we will develop and implement a regression loss with uncertainty for binary segmentation using bounding box labels. Due to the simplicity of regression with uncertainty, the purpose of this stage is to evaluate the behaviour of the uncertainty on bounding boxes. However, since a classification loss performs better on segmentation tasks, we will expand the idea to develop a classification loss with uncertainty, both for binary and multi-class segmentation.

Chapter 2

Literature study

This literature study gives an overview of some topics which are needed to better understand the approach followed in this thesis. The first section introduces the concept of weakly-supervised training and gives an overview of the different types of supervision. The second section gives an introduction to uncertainty in deep learning, and how this can be implemented in a deep neural network. The last section shows how bootstrapping can be used to improve coarse labels, and how recent research have leveraged this technique.

2.1 Weakly-supervised training

Typically, to train a deep learning network, or a machine learning algorithm in general, a large dataset with many training examples is used. A training example consists of a feature vector, which for semantic segmentation tasks usually boils down to the input image, and a corresponding ground-truth vector. The latter resembles the ground-truth for the output of the network given the input feature vector. As discussed in the previous section, obtaining the ground-truth labels can be difficult due to the large cost. Therefore, it is interesting to develop algorithms which can be trained through a weak supervision approach. [11] gives an abstract overview of the weak supervision types. In general, there are three types of weak supervision: incomplete supervision, inexact supervision and inaccurate supervision.

Incomplete supervision

Incomplete supervision means that the training dataset consists of a small set of labelled data and a large set of unlabelled data. The set of labelled data is insufficient to train a good model. To train a model with incomplete supervision there are two techniques: active learning and semi-supervised learning. Active learning assumes there is a way to label the unlabelled data, except this is a costly operation. The objective of active learning is to minimize the labelling cost, while maximizing the learning of the model. To achieve this, certain criteria of the unlabelled data like informativeness and representativeness has to be evaluated before deciding to label it. Semi-supervised learning leverages the unlabelled data to ameliorate the model without human intervention. The idea is that given the labelled training data, the unlabelled data can implicitly give information about the true distribution of the data which can be helpful for predictive modeling.

Inexact supervision

Inexact supervision is the case where labelled data is given, but is not exact as desired. For example coarse-grained label information. Techniques regarding this type of supervision

usually regards multi-instance learning.

Inaccurate supervision

Inaccurate supervision concerns labels which consists of erroneous information. Typically, this is denoted as label noise. Some approaches to tackle this type of supervision consists of complicated data-editing techniques. The objective is to find the erroneous data and make some sort of correction.

Knowing the types of weak supervision, training a deep neural network with bounding box images can be seen as inaccurate supervision. While a majority of pixels in bounding box labels are labelled correctly, pixels around objects are erroneous and will cause false predictions when these are not accounted for.

2.2 Uncertainty in deep learning

Standard deep learning networks for classification output normalized score vectors, without a measure of uncertainty. Knowing what a model does not know is very important. Relying blindly on the output of a model can have disastrous outcomes: the classification of the white side of a trailer for bright sky resulted in the first fatality, while another erroneous prediction raised concerns of racial discrimination due to the identification of two African Americans as gorillas. If only the model could indicate a measure of uncertainty for these predictions, unfortunate outcomes could be avoided.

In [4], this problem gets addressed by implementing a method which incorporates both aleatory and epistemic uncertainty to the network by using Bayesian modeling. Aleatory and epistemic are the two main types of uncertainty. Aleatory uncertainty captures noise inherent in the observations. For images, this can be blurred sections, poorly lighting, occlusions, etc.. Collecting more data cannot reduce this type of uncertainty. Epistemic uncertainty captures the uncertainty in the model parameters. This uncertainty can be seen as our ignorance about which model generated the collected data, and is therefore also referred to as model uncertainty. Epistemic uncertainty can be completely explained away given enough data. Aleatory uncertainty can be further divided in two categories: homoscedastic and heteroscedastic uncertainty. Homoscedastic uncertainty stays constant for different inputs. Heteroscedastic uncertainty depends on the inputs of the model: some inputs can have more heteroscedastic uncertainty then others e.g. blurred regions in the image. Therefore, heteroscedastic uncertainty can be valuable to computer vision. One example from [4] is depth regression: an image with strong vanishing lines would result in high confidence, while a featureless wall would have high uncertainty. [4] models the epistemic uncertainty by putting a prior distribution on the parameters weights, such as a Gaussian distribution (referred to as Bayesian model). Heteroscedastic uncertainty gets represented by an additional output value σ, which is made data-dependent and can be learned as a function of the data:

$$L(\hat{y}, sigma|y) = \frac{1}{2\sigma^2}(y - \hat{y})^2 + \frac{1}{2}\log\sigma^2 \tag{2.1}$$

In a standard neural network, σ is often fixed as part of the model's weight decay and ignored (see [3]). The loss function to learn heteroscedastic uncertainty can also be used without a Bayesian neural network. As the epistemic uncertainty can be explained away with enough data, we are only interested in the heteroscedastic uncertainty.

2.3 Bootstrapping

As already mentioned, current deep learning approaches are trained in a supervised fashion and require an annotated dataset. The performance of the network heavily depends on the accuracy of the annotations. Some of the images might have erroneous annotations such as missing annotatons, subjective labeling or in-exhaustively annotated images. Bootstrapping is a way to overcome this problem by iteratively improving the noisy training targets.

[7] is a recent study on this topic. The method described in this paper shows how bootstrapping can be used to train on a noisy dataset. The idea they propose is to dynamically update the training targets based on the current state of the model. The training target is a combination of the noisy label and the model's prediction. As the model improves, the prediction of the model can be trusted more. This way, the model can disagree with the label. Incorrect labels are likely to be eventually highly inconsistent with other stimuli which have the same label predicted by the model. The dynamically improving training labels allow the network to keep updating to a be a better classifier. Equation 2.2 shows the adapted cross-entropy loss function. Instead of only using the training target y, the new target is a weighted sum of both y and the model's prediction q.

$$L(q, y) = [\beta y + (1 - \beta)q] \log(q) \tag{2.2}$$

Chapter 3

Setup

3.1 Dataset

The Cityscapes dataset [1] is used for evaluating the different methods described in this thesis. Cityscapes is a large-scale dataset and benchmark suite for training and testing pixel- and instance-level semantic labelling approaches (see figure 1.2). Cityscapes focuses on urban scene understanding and has a large set of images and stereo videos recorded in over 50 different cities. The dataset contains 5000 high quality pixel-level annotations and another 20 000 images which are only coarsely annotated. The latter can be used to train and test approaches that leverage large volumes of weakly-labelled data. The dataset offers annotations for 30 different classes like cars, persons, sidewalks... To test the methods developed in this thesis we will only focus on the car and pedestrian classes.

Since the methods in this thesis are based on bounding box annotations, we will have to transform the binary object masks from the Cityscapes dataset into bounding box annotations. Therefore, a python script is implemented to transform the pixel-perfect annotations into bounding box annotations (see figure 3.1). Although we use bounding box annotations to train our methods, we still use the pixel perfect annotations for validation and testing. For our experiments, we construct a dataset consisting out of 3475 images following a 80-10-10 ratio for training, validation and testing.

3.2 Deep learning framework

We will use the PyTorch framework [6] to develop and test the new methods in this thesis. PyTorch is a deep learning framework using the python language and based on the Torch platform. The main advantages of PyTorch is its pythonic nature, meaning it is easy to use, and the fact that it uses a dynamic computational graph. The benefit of a dynamic graph is that the graph can change during runtime, in contrast to a static computational graph which remains fixed.

3.3 Network architecture

The goal of this thesis is to develop methods for training a semantic segmentation network based on weakly-annotated images. To this end, the focus is on the development of the loss function and not specifically on the network architecture. Therefore, we choose an existing network architecture which achieves good results on standard semantic segmentation tasks and keep the network fixed for all experiments. Favoring a good trade-off between speed and accuracy, we choose the ERFNet network [8, 9] (see figure 3.2). ERFNet's architecture

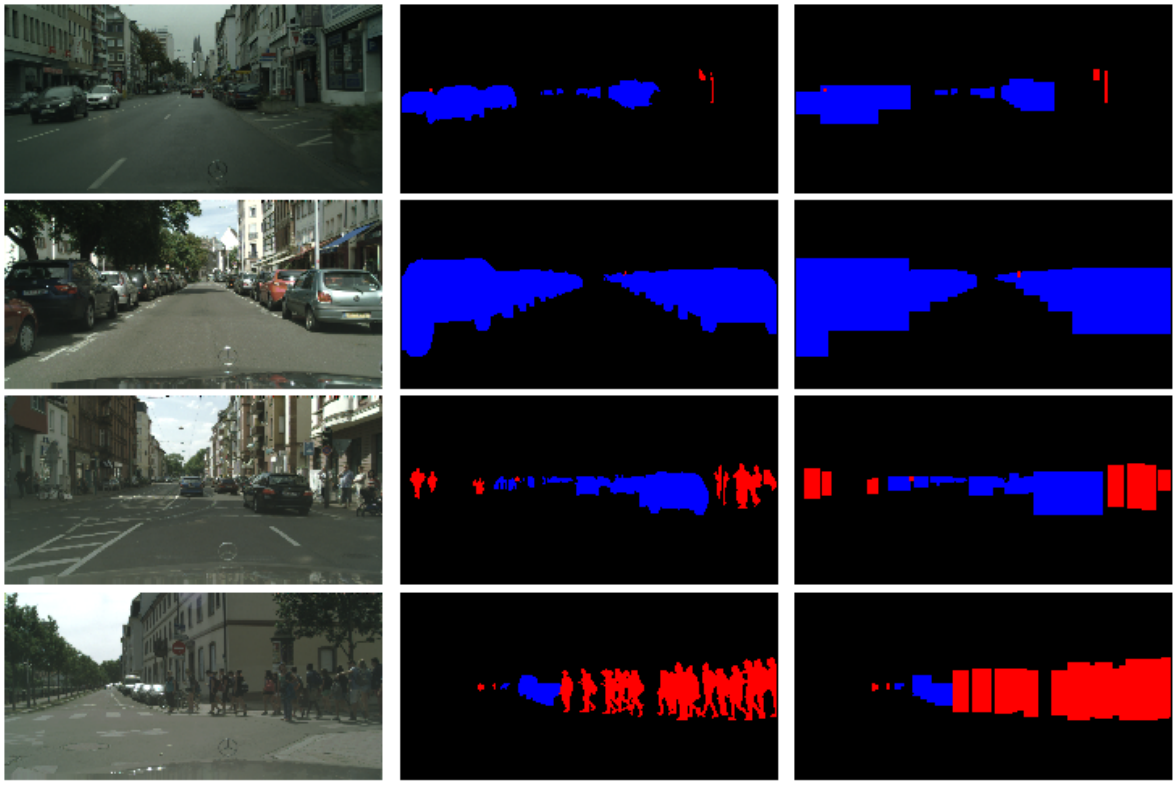


Figure 3.1: From left to right: image, ground truth label, bounding box label.

is optimized for real-time semantic segmentation. This is achieved by using a layer that uses residual connections and factorized convolutions. At the time of publication, ERFNet achieved results similar to the state of the art when tested on the Cityscapes dataset.

To train the network we adopt the settings used in [9]. The Adam optimization algorithm [5] for stochastic gradient descent is used with a weight decay of $2e$-4 and an initial learning rate of $5e$-4. The learning rate gets decreased by a factor of 2 every five epochs the loss does not improve, with a cooldown of five epochs. The batch size is 6. These settings have proven to be efficient for every experiment done in this thesis.

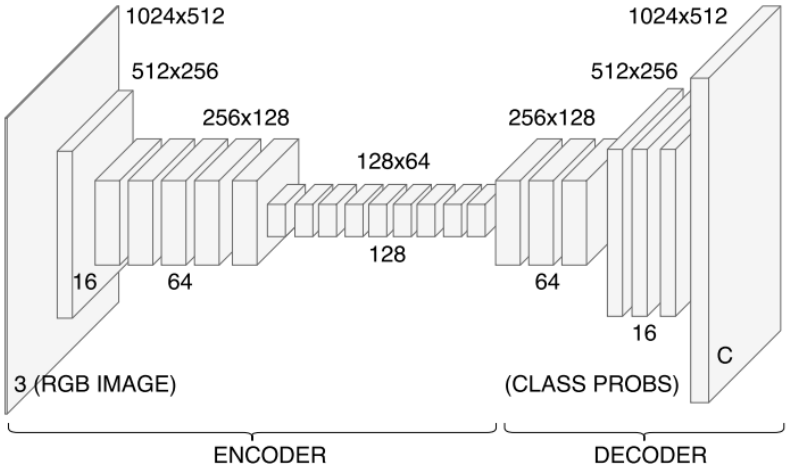


Figure 3.2: The figure displays the architecture of ERFNet [8].

Chapter 4

Binary segmentation without uncertainty

The goal of the experiments described in this chapter is to examine the segmentation results when training the network with only bounding box labels. To simplify these baseline experiments, we will only focus on the car class instead of using all classes labelled in the Cityscapes dataset. This way, the objective becomes binary segmentation: the network has to label pixels as foreground (car pixels) or background (all other pixels).

Since a bounding box around an object is only a crude estimate of the true object's mask, the bounding box will contain both back- and foreground pixels. Since all pixels within the bounding box are labelled as foreground, the network is forced to predict both as foreground (see figure 4.1).

The hypothesis is that the network is not capable of labelling background pixels within a bounding box as foreground, while labelling all other background pixels outside the bounding boxes as background, since they share the same statistics. As this behaviour is exactly what we are aiming for, a poor performing network would be a positive result, as the network would fail to label background pixels within a bounding box as foreground and label only the true foreground pixels as foreground.

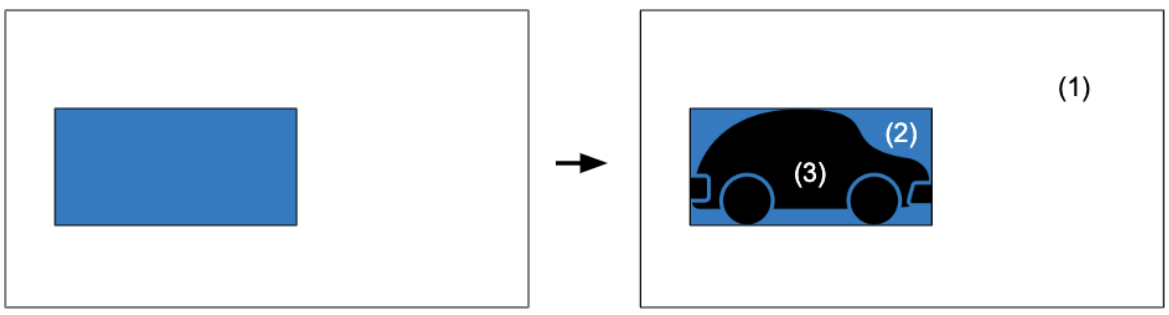


Figure 4.1: (a) An example of a binary bounding box label. (b) The true class labels: (1) background pixels labelled as background , (2) background pixels incorrectly labelled as foreground and (3) foreground pixels labelled as foreground.

(a). Input image

(a). Output prediction

Figure 4.2: Output of ERFNet trained with the standard BCE loss on bounding box labels. The result is a segmentation of nearly perfect rectangles around the objects. The network has no difficulties following the bounding box labels and can label background pixels withing the boundingbox as foreground.

4.1 Training with standard BCE loss

This experiment shows the result of ERFNet trained with only bounding box labels. The visual results are displayed in figure 4.2. Contrary to our hypothesis, the network is able to follow the boundingbox labels perfectly, resulting in a rectangle around each car.

The reason why the network is able to label background pixels within the boundingbox as foreground and the other background pixels outside the boundingbox as background is probably due to the large receptive field of the neural network. Although a large receptive field is one of the strong advantages of a deep convolutional neural network, in this case, due to the (incorrect) bounding box labels, it prevents the network to segment distinct car shapes. As explained in the previous section, the network is (incorrectly) forced to segment background pixels within a bounding box as foreground. Normally this is counter-intuitive for the network, but because of the large receptive field, background pixels within a large range around cars can still be labelled as foreground.

To illustrate the effect of the receptive field's size, a small experiment was performed using a network with a small receptive field (17 pixels). As the network only consists of a couple of convolution layers, it cannot perform as well as the "large" ERFNet used in the previous experiment. However, due to the size of the network, it behaves more as a low-level binary segmentation filter. Figure 4.3 shows the result. It is clear that when using a small receptive field, the network is unable to distinguish between background pixels within or outside the boundingbox and therefore cannot learn to segment rectangles around an object. This results in some clear edges around the cars. But, by limiting the size of the receptive field and so also the strength of a convolutional network, the result are rather poorly. Nevertheless, this experiment gives a good insight in the effect of the receptive field of the network, and could be an interesting topic for further research.

4.2 Removing the rectangular bias

The previous section showed that the use of bounding box labels resulted in a segmentation of perfect bounding boxes. One explanation for this is that the network fails to label background pixels within a bounding box as background. Since the labels used in the previous experiment only consisted of perfect enclosing bounding boxes, the network gets biased to predict perfect enclosing rectangles. To prevent this, the bounding box labels were adapted: instead of a perfect enclosure around the object, the size of the bounding

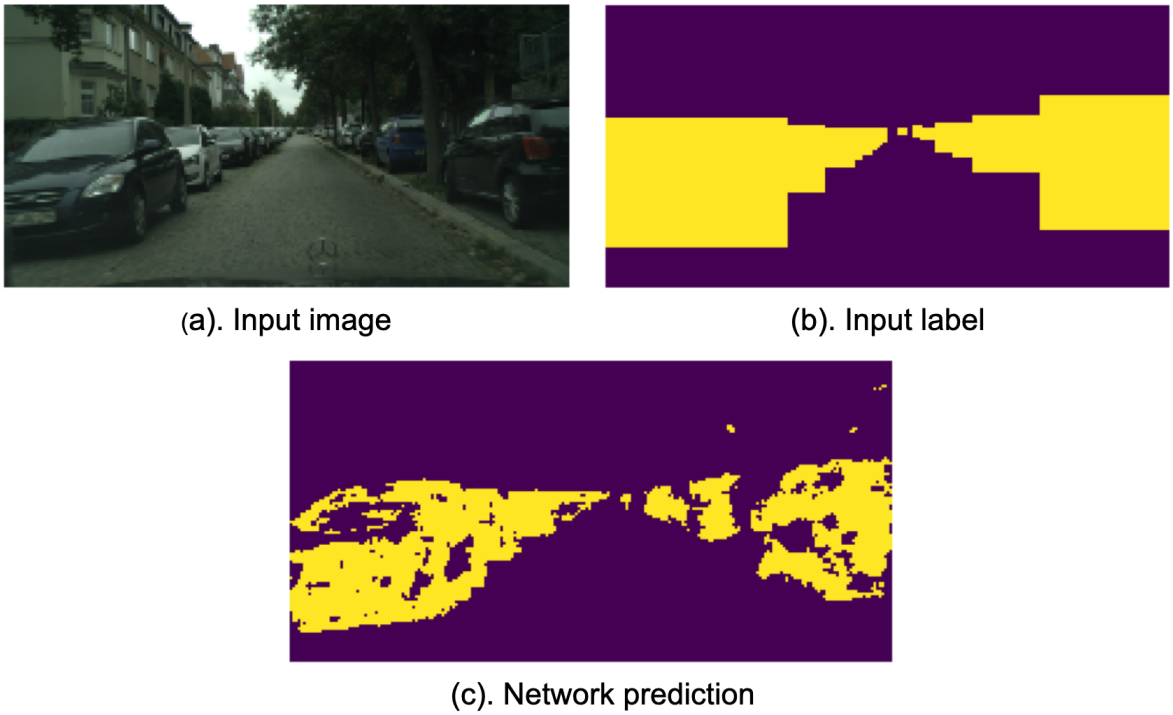


Figure 4.3: Results after training a CNN with a small receptive field. The output prediction clearly shows some distinct car shapes, indicating that a small receptive field prohibits the network to predict background pixels within a bounding box as foreground. But, due to the size of the network, the segmentation results are poorly.

box was randomly halved. This way, the pixels around cars are randomly labeled as fore- or background, removing the bias of a perfect rectangle. Figure 4.4 show the output of the network. The prediction now is a rectangle somewhere in between the perfect enclosing and the halved-size rectangle.

4.3 Attenuating the loss within the bounding box

Previous experiments showed that the use of bounding box labels forces the network to segment an enclosing rectangle around the object. The label forces each pixel within the bounding box to be labelled as foreground. This may be hard for the network to predict, because a large portion of the true background pixels have similar features. For this reason, the experiment described in this section uses a loss attenuation term within the bounding box. By attenuating the loss at certain locations, the network is given the opportunity of ignoring the groundtruth label. This way, the prediction of background pixels within a bounding box as background is not penalized as heavily. Hence, the network can freely predict the labels where the loss is heavily attenuated. As the background pixels within a bounding have similar features to the true background pixels, the expected result is for the network to segment only the true car pixels as foreground. The attenuation mask used in this experiment has the form of a gaussian mask within the bounding box. The effect of this form of attenuation is that the loss for the pixels at the edge of the bounding box get heavily attenuated as opposed to the pixels at the center. This because at the border of the boundingbox the labels show the most errors. The result can be seen in figure 4.5. The experiment shows that even an attenuation of the loss within the bounding box cannot prevent the network from predicting a perfect rectangle around each object.

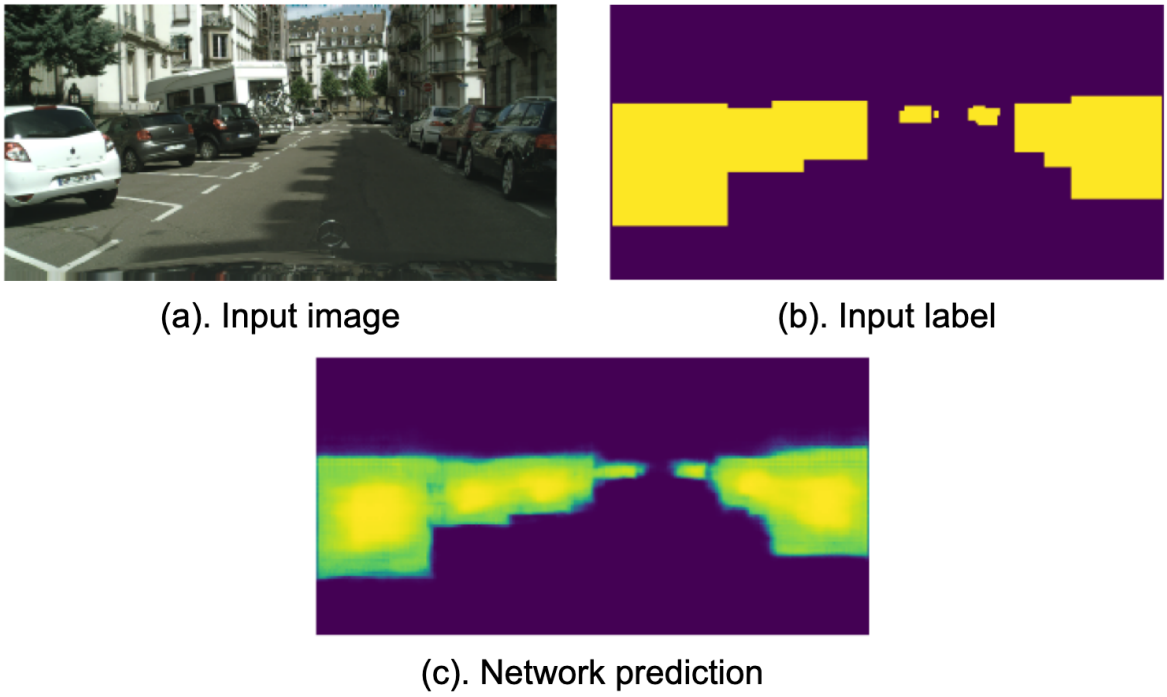


Figure 4.4: The figure shows the output prediction of a network trained with randomly halved bounding box labels. The output of the network is a prediction of a rectangle with a size somewhere in between the perfect enclosing and halved rectangle. The network still is not capable of segmenting any distinct car shapes. The prediction image is the result of the last sigmoid layer before applying a threshold.

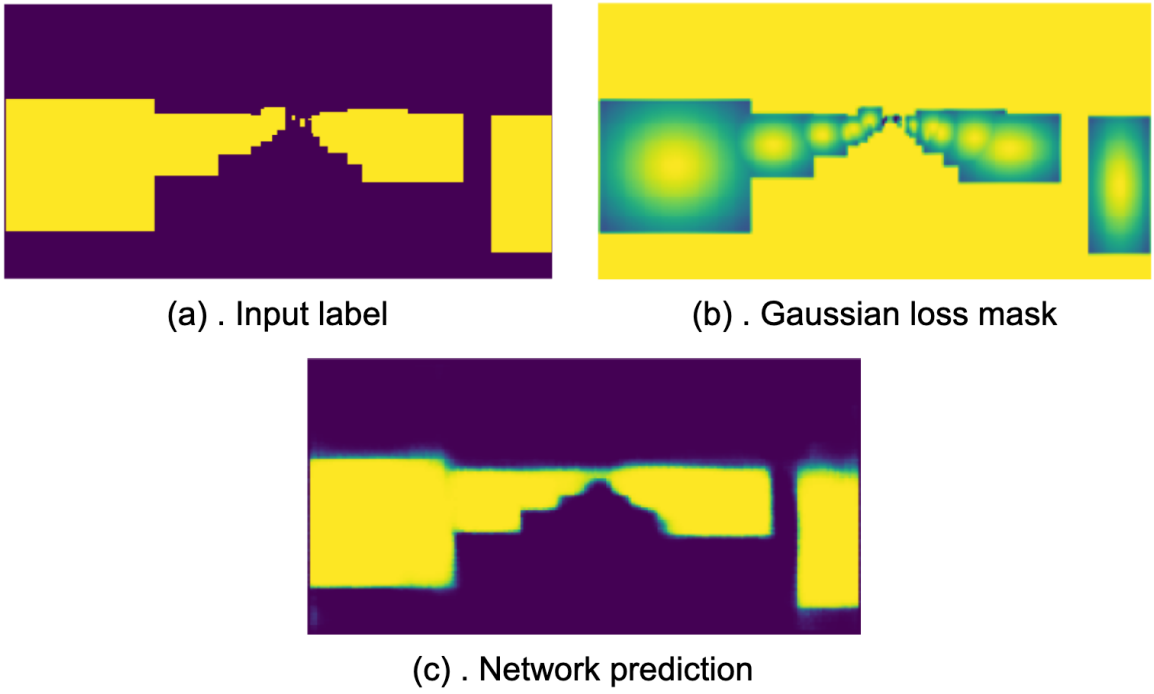


Figure 4.5: The figure shows an input label (a) with its corresponding Gaussian loss mask (b). The loss mask attenuates the loss at the border of the bounding boxes. As seen in the network's prediction (c), the attenuation of the loss has no effect on the prediction of perfect rectangles around the object.

4.4 Conclusion

This chapter shows that a standard CNN like ERFNet with a classification loss function like the BCE loss cannot be trained with bounding box labels to perform semantic segmentation. The result of such a setup is a segmentation of rectangles around the objects, which is not the desired. The different ad-hoc techniques used in this chapter to overcome this problem did not live up to expectations. The following chapters describe an implementation of a new loss function, which incorporates uncertainty, to train a semantic segmentation network with bounding box labels.

Chapter 5

Regression loss with uncertainty

This chapter will implement a regression loss function with uncertainty to train a binary segmentatino network with bounding box labels.

5.1 Regression loss with uncertainty

A standard regression loss is often used to infer a continuous variable like depth regression. Standard regression uses a fairly simple loss like an L1 or L2 loss.

$$L2\ loss = \sum_{n=i}^{N}(y - \hat{y})^2 \tag{5.1}$$

The L2 loss, also referred to as the mean squared error loss (figure 5.1), measures the squared difference (error) between the inferred value and the label. The objective is to minimize the squared error and fit a model to the data set. To introduce uncertainty to the L2 loss, a Gaussian likelihood is introduced [4]. Instead of inferring a single output value, the model now infers two values: $\hat{y}$ and σ^2. These values now represent a Gaussian probability function for the output, instead of a single most optimal value. The aleatory uncertainty in this model is represented by σ^2. The objective of the loss function is to maximize $P(y|\hat{y}, \sigma^2)$ with $P \sim \mathcal{N}(\hat{y},\ \sigma^2)$. As described in [4], the loss is derived as follows:

$$P(y|\hat{y}, \sigma^2) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}(\frac{y-\hat{y}}{\sigma})^2} \tag{5.2}$$

Which leads to:

$$P(y|\hat{y}, \sigma^2) = \frac{1}{2\sigma^2}(y - \hat{y})^2 + \frac{1}{2}\log\sigma^2 \tag{5.3}$$

As the uncertainty term directly learns from the loss function itself, it is completely unsupervised. The new loss function can be seen as the L2 loss, attenuated by a factor $\frac{1}{2\sigma^2}$ and penalized by $\frac{1}{2}\log\sigma^2$. However, the authors in [4] stress this is not an ad-hoc solution, but the result of the probabilistic interpretation of the loss function.

The uncertainty enables the network to deal with noisy labels. The network can predict values differing from the label by increasing it's uncertainty. Normally, the L2 loss would result in a squared error of any difference from the inferred value and the label. By increasing the uncertainty, this error can be lowered through the attenuation term. The penalty term can be seen as a regularization of the uncertainty, limiting the attenuation term not to grow infinitely large for every pixel as seen in figure 5.2.

The loss function described in this section can be used to perform binary classification. However, while generally a regression loss performs worse than a standard classification

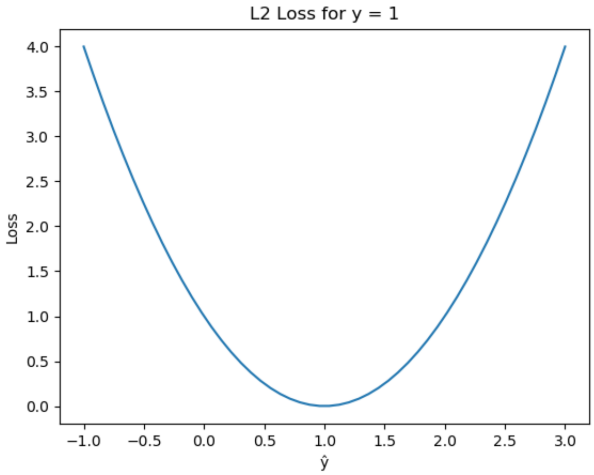


Figure 5.1: The L2 loss function.

loss like the cross-entropy loss function, the use of a regression loss with uncertainty is much simpler to implement than a classification loss with uncertainty. For this reason, we will first test the regression loss to train a segmentation network with bounding boxes, before developing and implementing a classification loss.

5.2 Binary segmentation with uncertainty

The regression loss function with uncertainty (eq. 5.3) was tested by training a model with the bounding box data. The result can be seen in figure 5.3. Again, the model fails to label background pixels within the bounding box as background. The initial hypothesis was, using the uncertainty loss, the background pixels within the bounding box would get labeled as background contrary to its label, but with high uncertainty. The uncertainty would attenuate the loss function, so the model could label pixels differently than the corresponding label. However, the result in figure 5.3 shows that the output still is a perfect rectangle. Only a small border around the rectangle has high uncertainty, indicating a high "difficulty" at the border of each rectangle.

A closer look at the uncertainty loss function

Figure 5.2 shows the uncertainty regression loss function for a pixel with label y = 1, while the inferred value $\hat{y}$ is 0, 0.25 and 0.5. These three functions give a good insight in how the function actually works. When an inferred value $\hat{y}$ strongly differs from the label y, the loss function can either be minimized by increasing the variance or by moving $\hat{y}$ closer to the label. However, both cases do not have the same gradient. The gradient on $\hat{y}$ is much higher than the gradient on the variance. For example, the function in figure 5.2 having $\hat{y} = 0$ can only be lowered to around 0.5 by changing the variance to its optimal value. Changing $\hat{y}$ closer to 1 will result in a much larger drop in the loss function as seen in figure 5.4. With this in mind, the result of previous experiment can be explained by this reason. Even if the model wants to label background pixels within a bounding box as background while increasing the uncertainty, the reward for following the bounding box labels is much higher due to the larger drop in the loss function.

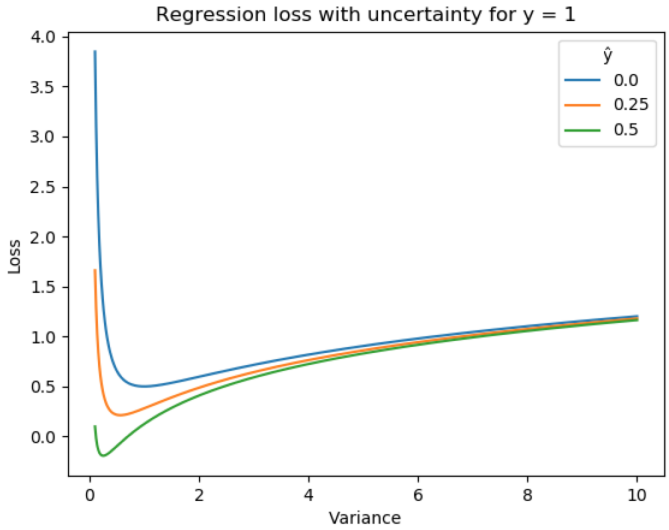


Figure 5.2: The regression loss function with uncertainty for y = 1 (eq. 5.3). The graphs depicts three different inferred values for $\hat{y}$. As $\hat{y}$ is further away from the label y, the minimum of the loss function rises. This is due to the l2 term in the loss function. Increasing variance results in a decreasing loss function, up to a certain point, where the penalty term takes over and increases the loss function. Nevertheless, the model can lower the loss function to 0.5 for an inferred $\hat{y}$ of 0, whereas a standard l2 loss would result in a value of 1.

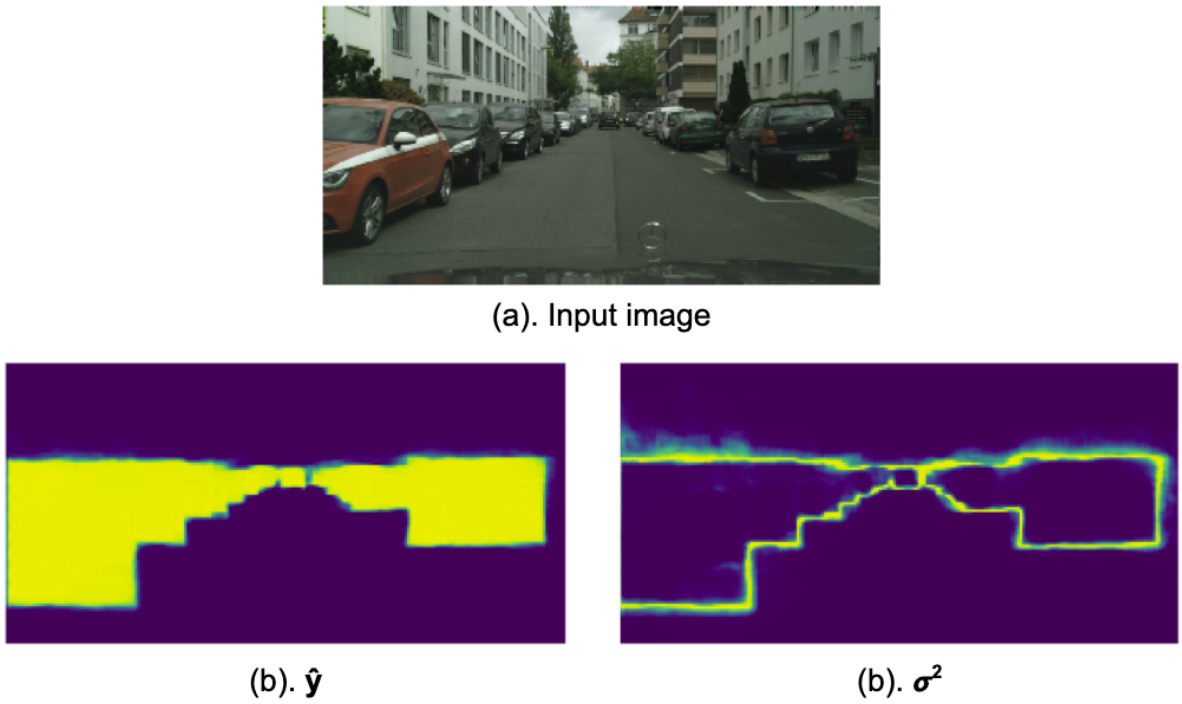


Figure 5.3: Output of a model trained with the regression loss function with uncertainty (eq. 5.3) using only bounding box labels. The figure shows the input image (a), the inferred mean $\hat{y}$ and σ^2, the aleatory uncertainty. The result is a segmentation of a perfect rectangle, with high uncertainty on the border.

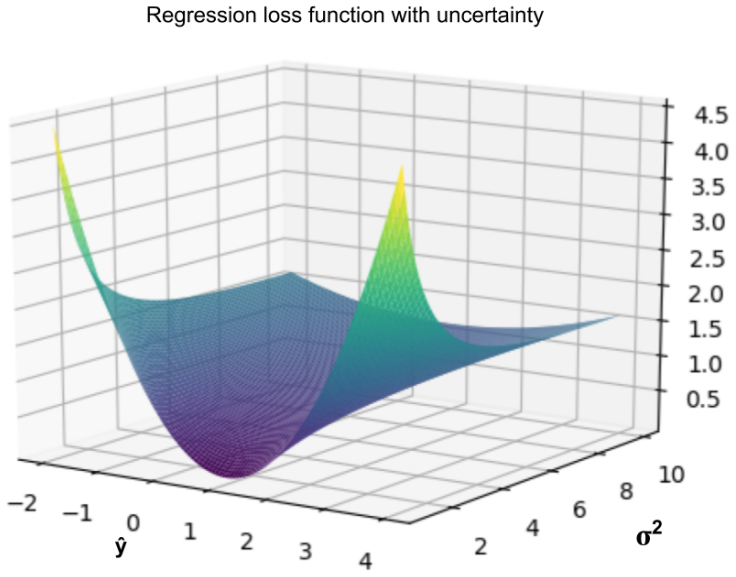


Figure 5.4: The figure shows the regression loss function with uncertainty. It is clear that the gradient on $\hat{y}$ is much larger than the gradient on σ^2.

Feeding in ground truth labels

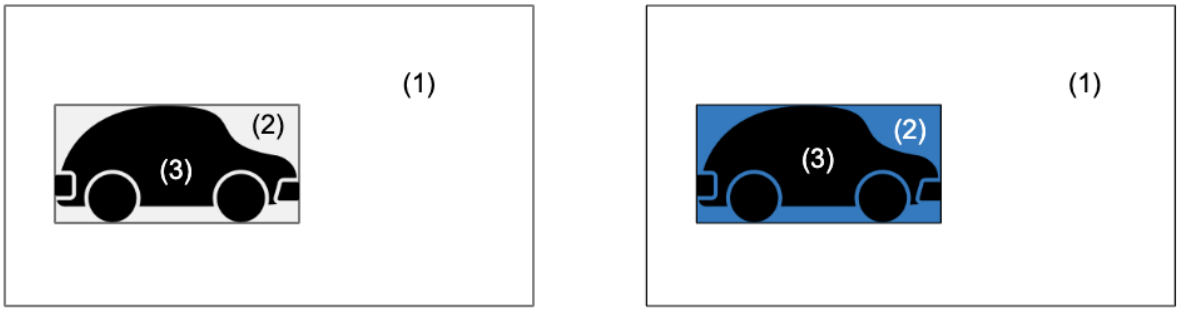
Since there is no reason for the model to infer pixels opposite to their label value, a solution might be to mix in some ground truth labels in the data set. A training batch now consists of both ground truth labels and bounding box labels. Since we are certain of the validity of the ground truth labels, the loss function used on these images is a normal regression loss. The batch loss now consists of both the normal regression loss summed with the uncertainty regression loss. Figure 5.5 shows an abstract version of a ground truth and bounding box label. The batch loss for pixels near a car are now a sum of the standard regression loss with label y = 0, and the uncertainty regression loss with y = 1. Minimizing the loss for the ground truth labels results in predicting $\hat{y} = 0$. However, this results in a high uncertainty regression loss since the label for those pixels is y = 1. As in the previous experiment the model had no incentive to increase the uncertainty because of the higher gradient on $\hat{y}$, now, to minimize both the standard regression loss for the ground truth labels as well as the uncertainty regression loss for the bounding box labels, the only way the minimize the summed loss is to increase the uncertainty while keeping $\hat{y} = 0$. Equation 5.4 describes the loss function for pixels near a car which are labeled as background in ground truth labels and as foreground in bounding box labels. The first term of the loss can be minimized only by setting $\hat{y}$ as 0. Having $\hat{y} = 0$, the second term must be minimized by increasing the uncertainty.

$$
\begin{aligned}
L &= L(GT) + L(BBox) \\
&= (y - \hat{y})^2 + \frac{1}{2\sigma^2}(y - \hat{y})^2 + \frac{1}{2}\log\sigma^2
\end{aligned}
$$

Substituting y = 0 for ground truth labels and y = 1 for bounding box labels

$$
= (0 - \hat{y})^2 + \frac{1}{2\sigma^2}(1 - \hat{y})^2 + \frac{1}{2}\log\sigma^2 \tag{5.4}
$$

The experiment was done by using 500 ground truth labels and 2280 bounding box



(a) Ground truth label (b) Bounding box label

Figure 5.5: The figure shows the different sections in a labeled image. For the bounding box label: background pixels labeled background (1), background pixels labeled foreground (2) and foreground pixels labeled foreground. The ground truth pixels does not have wrongly labeled pixels.

labels. First, a model was trained using the standard regression loss with the 500 ground truth labels. Next, the model was trained by combining the ground truth and bounding box labels together with the uncertainty regression loss. The result can be seen in figure 5.6. Feeding in the ground truth labels forced the model to increase its uncertainty at the wrongly labelled parts, because of the global minimum of the loss described in the previous paragraph. This way, the output of the model is now a decent segmentation of the car shapes, and does not show the rectangular bias anymore. In table 5.1 we can see that by training the network on an extra 2280 bounding box labelled images, the IoU score slightly improves from 69.43 to 79.02 IoU.

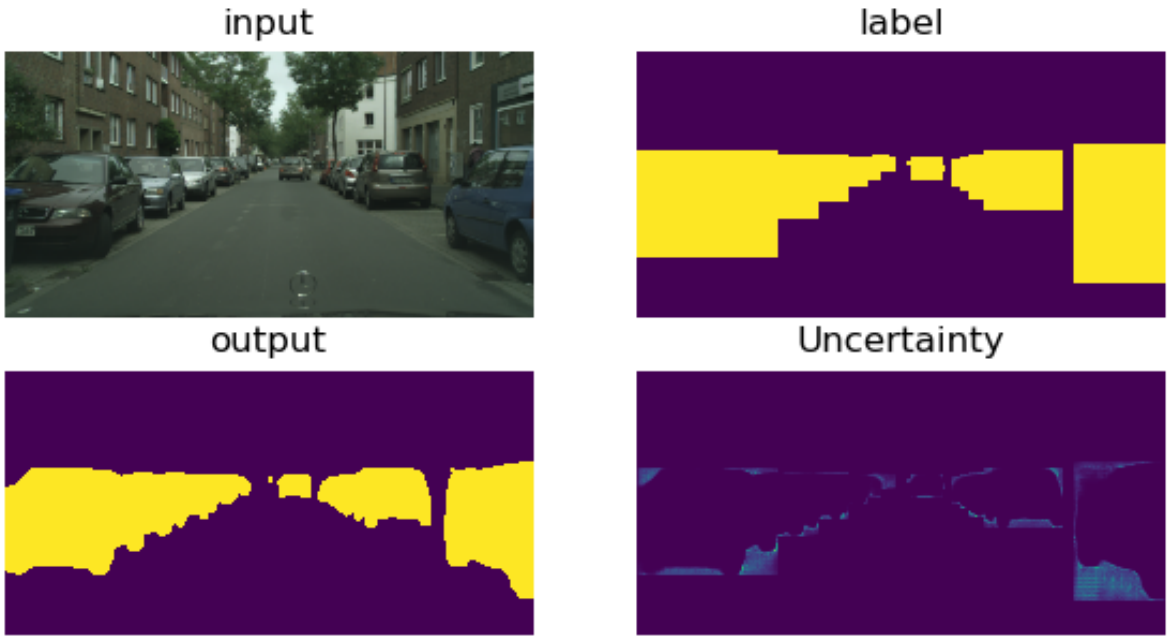


Figure 5.6: Model trained with the loss function specified in equation 5.3. By feeding in a small portion of ground truth labels, the model has learned to increase its uncertainty around each car. This way the model is able to label pixels within a bounding box as background without being penalized by the wrongly annotated bounding box labels. The output image of $\hat{y}$ depicts clearly the car shapes.

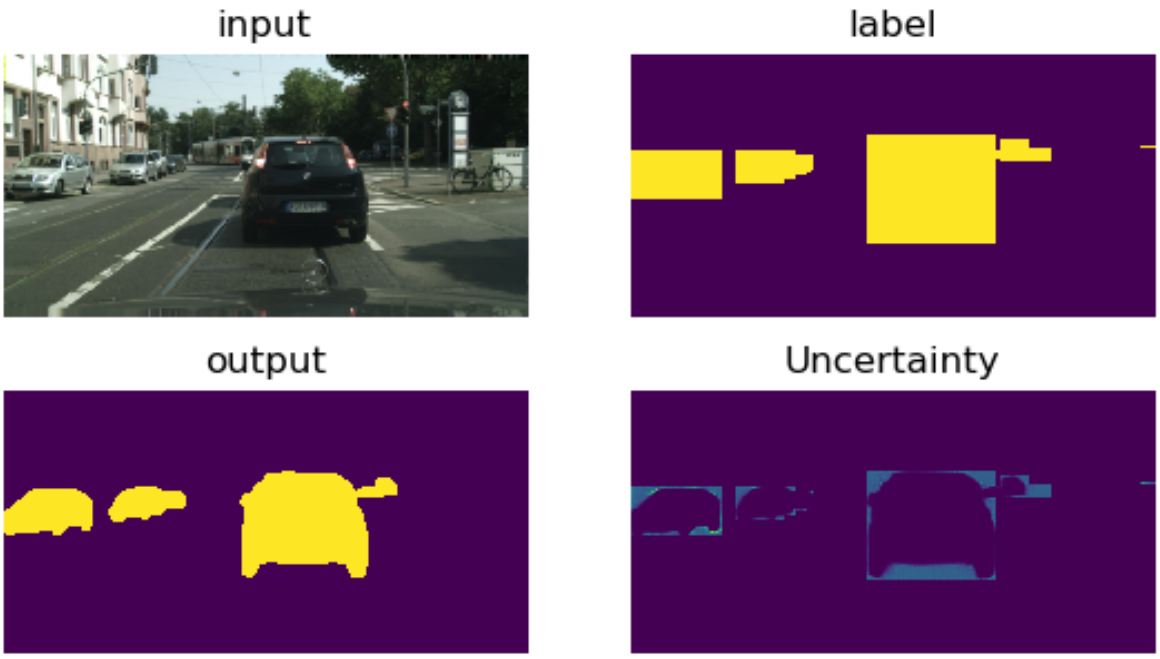


Figure 5.7: Output of a model trained with the regression loss function being applied only within the bounding box. Background pixels are scored by the regular MSE loss.

	Dataset		IoU (%)
	GT	***BBox***	***Car***
Standard BCE	2780	/	86.9
	500	/	78.6
	500	2780	**69.43**
Regression w Uncertainty	500	2280	**79.02**
Regression w Uncertainty (bbox only)	500	2280	**83.04**

Table 5.1: Regression with uncertainty results

Limiting uncertainty to within the bounding box

The uncertainty has the advantage of attenuating the loss when the network infers a value opposite to its ground truth label. As the bounding box labels only have wrongly labeled pixels within the bounding itself, there is no need to learn uncertainty outside these bounding boxes. Therefore by limiting the uncertainty regression loss (eq. 5.3) to only the inside of the bounding boxes, the standard regression loss (without uncertainty) can be used on the background pixels (output in figure 5.7). In table 5.1 we can see that this modification improves the IoU score from 79.02 to 83.04 IoU, which is an improvement of almost 4 IoU points compared to training the network on 500 groundtruth images only.

Chapter 6

Binary cross-entropy loss with uncertainty

The previous chapter made use of a regression loss for binary segmentation. Typically, a regression loss is not used for classification tasks and will (normally) perform worse. As mentioned in the previous chapter, we tested a regression loss with built-in uncertainty first, because of the simplicity of the implementation and to investigate the interaction between the use of bounding box labels and the uncertainty. In this chapter, we will implement and test the classification loss with uncertainty from [4].

6.1 Binary semantic segmentation with uncertainty

6.1.1 Standard BCE loss

The loss function commonly used for binary classification is the binary cross-entropy loss function (BCE loss). The last layer of the network outputs a logit value l, which is the logarithm of the odd ($\log(\frac{p}{1-p})$). The logit can range from $-\infty$ to $+\infty$. It gets squashed through the sigmoid function (eq. 6.1) to get a value between 0 and 1, representing the probability $\hat{p}$ of that pixel being 1 (which can be derived by inserting the logarithm of the odd into the sigmoid). The BCE loss function is described in equation 6.2. The loss function measures the cross-entropy between the probability of the network's output $\hat{p}$ and the probability of the label p. Figure 6.1 shows how a logit value gets squashed through a sigmoid to get $\hat{p}$. Next, this value gets through the BCE loss to get the loss value.

$$sigmoid(x) = \frac{1}{1 + e^{-x}} \tag{6.1}$$

$$L(\hat{p}, p) = p \log \hat{p} + (1 - p) \log(1 - \hat{p}) \tag{6.2}$$

6.1.2 Adding uncertainty to the BCE loss

To add uncertainty to the BCE loss, we use the implementation of the cross-entropy loss with aleatory uncertainty from [4]. Like the regression loss in the previous chapter, the output of the network get split in two. Instead of outputting a single logit value l, the network outputs a mean μ and variance σ^2: $logit \sim \mathcal{N}(\mu, \sigma^2)$. Whereas in the standard BCE loss, the logit gets squashed through the sigmoid to get the probability $\hat{p}$, we have to compute $\mathrm{E}[\hat{p}]$, the expected value of the probability:

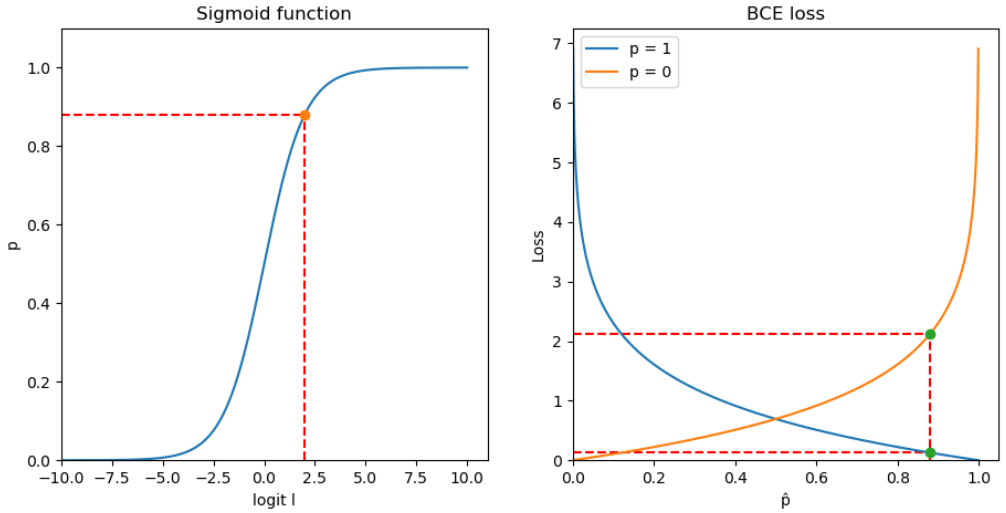


Figure 6.1: The left graph shows the sigmoid function which transforms a logit value l to the probability $\hat{p}$. The right graph depicts the BCE function, which takes the value $\hat{p}$ and calculates the loss, depending on the label p.

$$E[\hat{p}(l)] = \int sigmoid(l) * P(l)dl$$

With P(l):

$$P(l) = \frac{1}{\sigma\sqrt{2\pi}} \exp^{-\frac{1}{2}(\frac{l-\mu}{\sigma})^2} \tag{6.3}$$

Solving the integral to get the expected probability can be done by using Monte Carlo integration: logit values get sampled from the Gaussian distribution outputted by the network, squashed through the sigmoid function and averaged. Luckily, [10] approximated the integral:

$$E[\hat{p}(l)] = sigmoid(\frac{\mu}{\sqrt{1 + \pi\sigma^2/8}}) \tag{6.4}$$

The expected probability can be computed by first computing an "adapted" logit value from μ and σ^2 and taking the sigmoid function to get E[$\hat{p}$]. The loss value is the standard BCE of E[$\hat{p}$]:

$$L(\hat{p}, p) = p \log E[\hat{p}] + (1 - p) \log(1 - E[\hat{p}]) \tag{6.5}$$

Figure 6.2 shows the new loss function. The right graph shows how the loss gets lowered for increasing σ^2, while μ is opposite to the label p (smaller than 0 for $p = 1$, which will result in $\hat{p} < 0.5$). When μ is according to the label (greater than 0 for $p = 1$), the loss will increase by increasing σ^2. The limit of both cases will diverge to $-\log 0.5$. This is caused by the sampling of the logits through the sigmoid function seen in figure 6.3. A large value of σ^2 will result in the sampling of an equal amount of sigmoid values of 0 as of 1, which averages to 0.5.

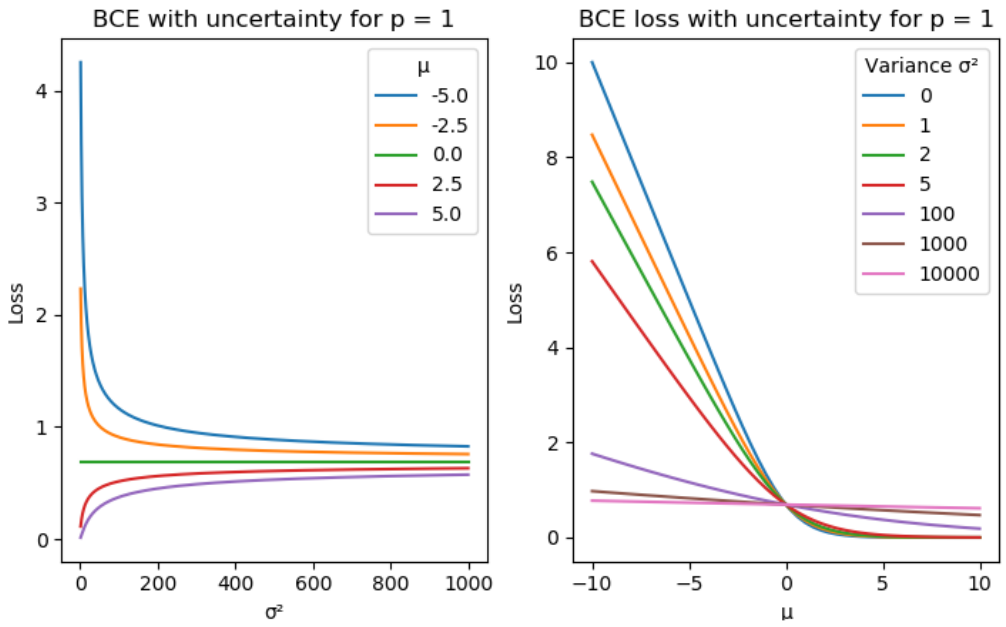


Figure 6.2: The left graph shows the BCE loss with uncertainty in function of σ^2 with label p = 1. It is clear that μ values, smaller than 0 (which results in $\hat{p} < 0.5$) will have a lower loss with increasing σ^2. The loss will rise for μ values greater than 0, because these μ values result in $\hat{p} > 0.5$. Increasing variance will result in the divergence of the loss to $-\log(0.5)$. This is because of the sampling of the sigmoid seen in figure 6.3. The right graph plots the BCE with uncertainty in function of μ. Increasing variance will decrease the loss when μ is smaller than 0, while increasing the loss when the logit is greater than 0. This graph also depicts the divergence of the loss to $-\log(0.5)$ for large values of σ^2

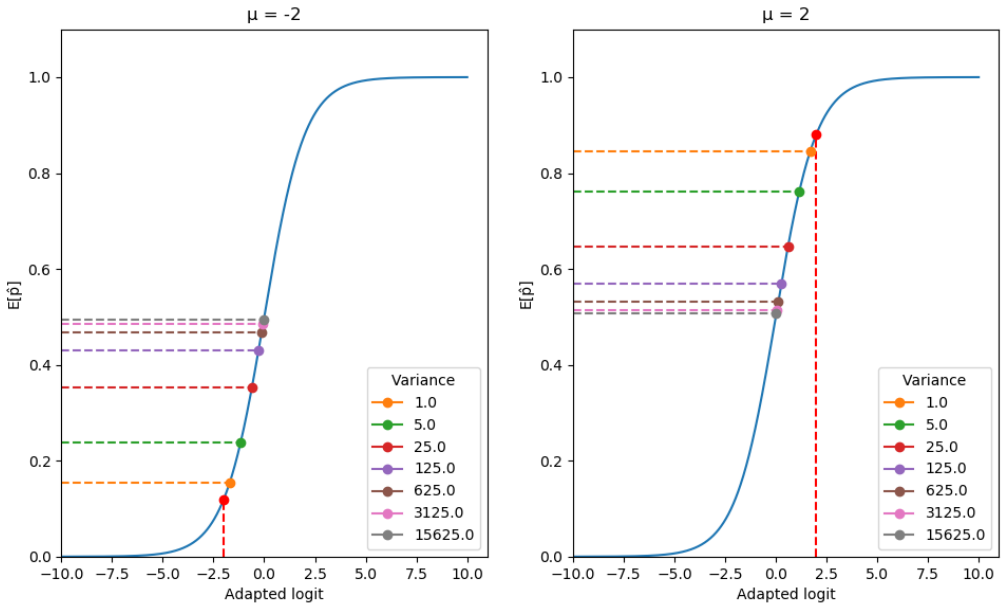


Figure 6.3: The figure shows the expected sigmoid value for both μ values of -2 and 2. Since $logit \sim \mathcal{N}(\mu, \sigma^2)$, as the variance increases, the expected sigmoid value shifts towards 0.5. A large variance causes many logit samples to have a sigmoid value of 0 and 1, which averages to 0.5.

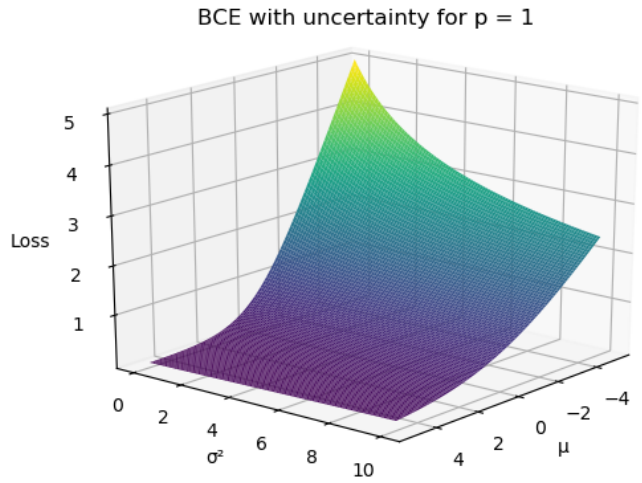
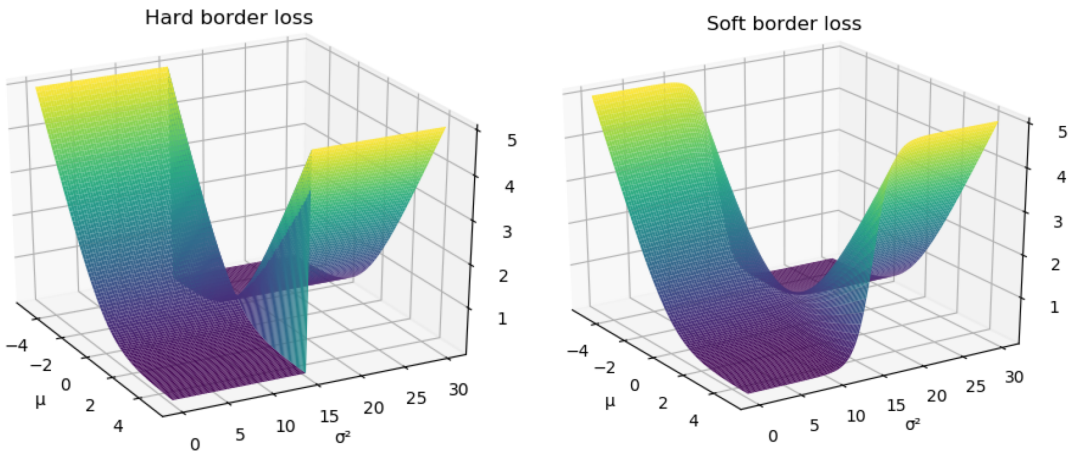


Figure 6.4: The figure shows a 3D plot of the BCE loss with uncertainty. When μ is opposite to the label (smaller than 0 when p = 1), a larger drop in the loss can be achieved by changing μ. The gradient on σ^2 is much smaller. Therefore, a model would label a background pixel within a bounding box as foreground rather than labelling it as background while increasing σ^2.

6.2 Problems when training on bounding box labels

Training a model using loss function 6.5 on only bounding box labels will not work right out of the box. The same disadvantage as with the regression loss with uncertainty from the previous chapter are the gradients (see fig. 6.4). To recall, equation 6.5 takes two values as input: μ and σ^2. This can be seen as a Gaussian distribution for the logit ($logit \sim \mathcal{N}(\mu, \sigma^2)$). When μ is in favour of the label p, increasing variance will increase the loss. When μ is opposite to the label p, increasing variance will result in a decreasing loss, up to a minimum of $-\log(0.5)$. When training a model on bounding box labels, the objective of the model is to label background pixels within a bounding box as background (while the label for these pixels is foreground). This means the network has to output μ opposite to the label, and decrease its loss by increasing the variance. Unfortunately, the gradient on μ is much larger than the gradient on σ^2. Changing μ towards the label p will cause a larger drop of the loss than changing σ^2. The result is a segmentation of bounding boxes.

The way we dealt with this problem in the previous chapter was to add some ground truth labels to the training set. To minimize the loss on the ground truth labels, the model has to label pixels near a car as background. To minimize the loss on bounding box labels, pixels near a car within the bounding box has to be labelled as background, while increasing the uncertainty. While this was a good solution for the regression case, this does not work for the BCE loss. Since increasing the uncertainty can only drop the loss to $-\log(0.5)$, the model favours to output bounding boxes, as this results in a lower loss.



(a) Hard border

(b) Soft border

Figure 6.5: The BCE loss for label p = 1. Figure (a) depicts flipping of the label on a threshold value of σ^2. Figure (b) has a more gradually transition from p = 1 to p = 0.

6.3 Flipping labels

To overcome the problem with the gradient described in the previous section, we introduce the concept of flipping labels (a binary version of bootstrapping). Because the gradient on μ of the pixels within the bounding box is always in the direction of foreground, the model is forced to update its parameters in this direction. Even for background pixels within the bounding box. By allowing the labels to flip, the gradient can change in the other direction. For example, a background pixel within a bounding box will get a steep gradient for μ towards foreground. If the label was flipped, the gradient would switch to background, enabling the loss to update the parameters of the model to label the pixels as background.

Labels are flipped in function of σ^2. If σ^2 is high, μ is opposite to the label. Still, μ gets a gradient towards the training label. For this reason, we introduce a second loss function which is the standard BCE loss but with a flipped label. Now the total loss function is the sum of the BCE with uncertainty for the training label and the standard BCE for the flipped label. There are two options when to decide to flip a label: an uncertainty threshold or a weighing of two loss functions for both labels as seen in figure 6.5. The summed loss can be seen in figure 6.6. The figure shows how the gradient changes when the uncertainty is high. The direction of the gradient on μ for the BCE with uncertainty is always towards the training label in contrast to the summed loss. Because of the addition of the normal BCE with the altered label, high σ^2 causes the label to flip, which changes the direction of the gradient. Because of the summation, the gradient towards the training label from the BCE with uncertainty is still present. For this reason, the BCE with uncertainty only gets differentiated with respect to σ^2. This enables the BCE loss with uncertainty to only learn σ^2. The standard BCE loss gets derived with respect to μ, enabling μ to only learn from this loss.

Figure 6.6 shows the summed loss function for label p = 1. When μ is -4, the gradient on μ is much larger than the gradient on σ. Figure 6.7 shows both parts of the summed loss. Since the BCE with uncertainty only gets differentiated with respect to σ^2, the only way the model can minimize this loss is to increase the uncertainty. While increasing σ^2, the label gets flipped for the second loss term. This causes to gradient to flip and enables the model to learn μ opposite to its training label. Because of the summation of the two loss terms, the summed loss has two optima. The model can label the pixel according

to its training label resulting in $\sigma^2 \inf 0$. Or the model can label the pixel opposite to its training label. This resulted in a high loss with a gradient on μ towards the training label (figure 6.7a when using only the BCE with uncertainty. Because of the flipped label when σ^2, the summed loss has a gradient in the opposite direction on μ.

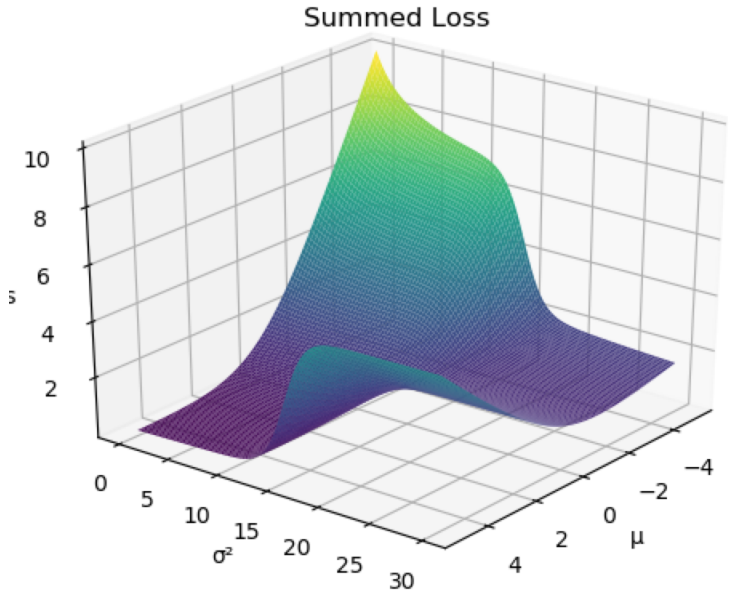
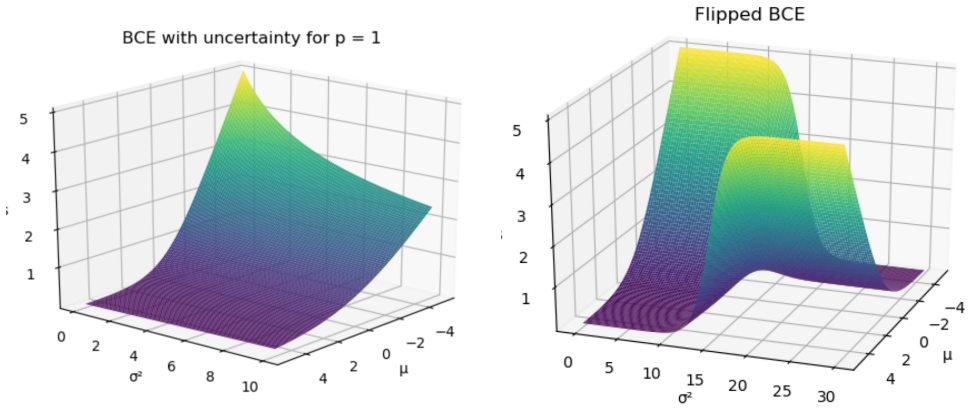


Figure 6.6: The loss shown in the figure is the sum of the BCE loss with uncertainty (figure 6.4) and the BCE with the label flipped based on the uncertainty (figure 6.5).



(a) BCE with uncertainty

(b) BCE with flipped label

Figure 6.7: The two parts of the summed loss. The BCE with uncertainty (a) gets differentiated with respect to σ^2. The BCE with the label based on σ^2 gets differentiated with respect to μ.

6.3.1 To summarize

Figure 6.8 shows a summary of the combined loss function for a training batch. A training batch consists of two types of labels: ground truth (which are pixel perfect annotated) and bounding box labels. The standard BCE loss is used for the ground truth labels to force the model to label pixels near objects as background. The bounding box labels are scored with two loss functions: the background pixels get scored by the standard BCE

loss (no need to train uncertainty, since these pixels are labeled correctly) and the pixels within bounding boxes are scored by our new loss function.

Our loss function is a sum of two terms: the uncertainty loss which trains the aleatory uncertainty (eq. 6.5) and the bootstrapping loss, which takes as input an altered training which can be flipped to background based on the uncertainty σ^2. The first term is only differentiated with respect to σ^2. The second term is differentiated with respect to μ

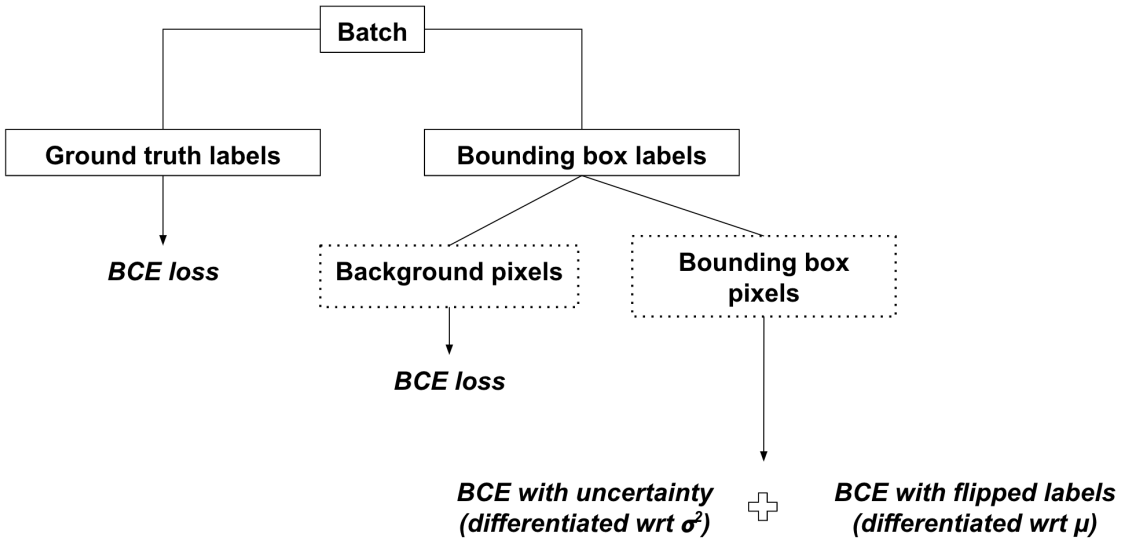


Figure 6.8: Representation of the combined loss for a batch consisting of both bounding box and ground truth labels.

.

6.4 Results

Table 6.1 shows the result of the experiments done with the loss function displayed in figure 6.8. The table shows the baseline IoU for cars for 2780 and 500 images trained with only ground truth labels. We see that the IoU of a dataset comprising of 500 ground truth and 2280 bounding box images is close to the baseline. The IoU drops when the amount of ground truth images is smaller. When the amount of ground truth images is 100, the IoU is still higher than the baseline of 500 GT images, but the segmentation of cars is converted to rectangles again. This is due to the effect of the gradient described in previous sections. Figure 6.9 shows an example of the output of the network. The figure also displays the flipping map, where the uncertainty is higher than the threshold value.

	Dataset		IoU (%)
	GT	BBox	
Standard BCE	2780	/	86.9
	500	/	78.6
	/	2780	69.18
	500	2280	**69.43**
Loss With Uncertainty	500	2280	**85.45**
	400	2380	85.32
	300	2480	85.42
	200	2580	84.37
	100	2680	81*

Table 6.1: The table shows the results for both the standard BCE loss and the classification loss with uncertainty developed in this chapter. (*) The output of this experiment resembles rectangles and is not the expected segmentation result.

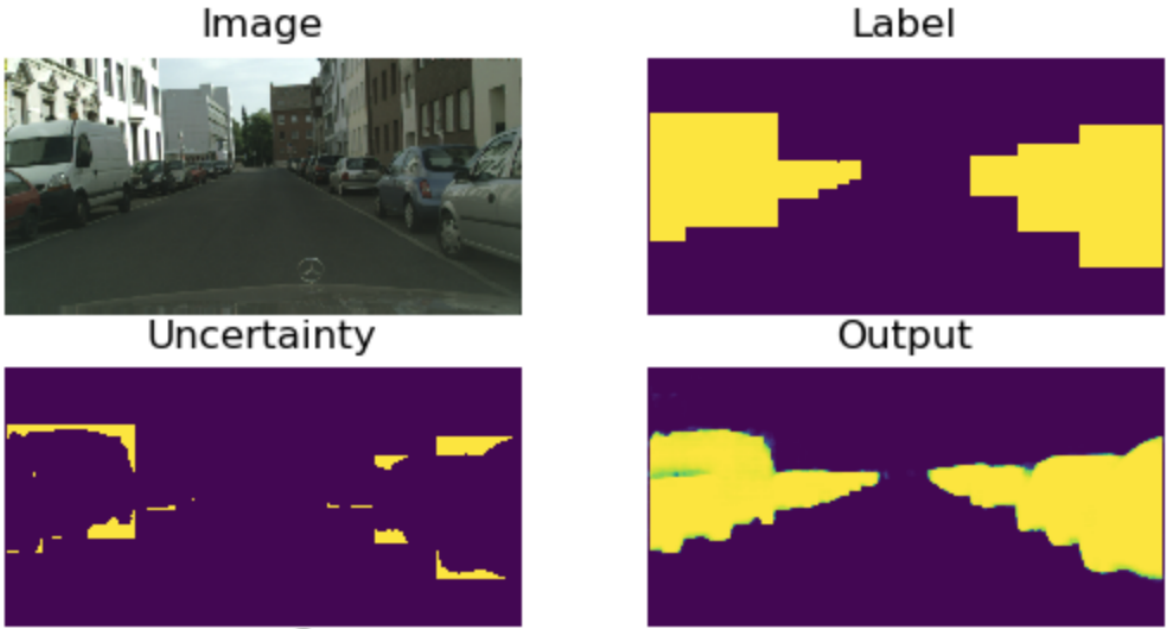


Figure 6.9: Output of the network trained with loss function 5.4. The uncertainty map displays which pixels have an uncertainty higher than the threshold, meaning the target gets flipped to background.

Chapter 7

Multi-class cross-entropy loss with uncertainty

The previous chapters describe binary segmentation with uncertainty. This chapter extends the ideas of the previous chapter to multi-class segmentation.

7.1 Cross-entropy loss with uncertainty

The standard loss function used for classification tasks is the cross-entropy loss (eq. 7.1). The cross-entropy loss measures the difference between two discrete probability distributions. Since there is a discrete set of classes, the class is a discrete random variable. The training label p is a vector of length C which sums up to 1, and represents a discrete probability distribution. The objective of the model is to output a distribution as close to the labels. The output of the network is a vector of logits. This vector gets squashed through the softmax function to get a probability vector which sums up to 1. The cross-entropy loss measures the distance between this probability vector $\hat{p}$ and the label vector p. Since the label vector has all its probability in one class ($p_y = 1$), the loss gets simplified to equation 7.2. The loss is the negative logarithm of the probability of the label class.

$$L = -\sum_{c=i}^{C} p_c \log(\hat{p}_c) \tag{7.1}$$

$$L = -p_y \log(\hat{p}_y) \tag{7.2}$$

7.1.1 Adding uncertainty

To add uncertainty, [4] proposed to split the output of the network in two: instead of outputting a single logit vector $\mathbf{l}$, the network outputs two vectors of length C. The output vectors represent logit means and variances ($\boldsymbol{\mu}$ and $\boldsymbol{\sigma^2}$). The logit vector now follows a Gaussian distribution: $\boldsymbol{l} \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\sigma^2})$. This changes loss function 7.2 to:

$$L = -p_y \log(E[\hat{p}_y]) \tag{7.3}$$

$$E[\hat{\boldsymbol{p}}] = \int softmax(\boldsymbol{l}) * P(\boldsymbol{l}) d\boldsymbol{l} \tag{7.4}$$

However, there is no solution to solve the integral analytically. The expected probability vector can be estimated through Monte Carlo integration. T sample logit vectors get

drawn from the Gaussian distribution outputted by the model, squashed through the softmax function and averaged. The equations below show a numerically stable approach from [4]:

$$\boldsymbol{l_t} = \boldsymbol{\mu} + \boldsymbol{\sigma} \odot \boldsymbol{\epsilon_t},\ \boldsymbol{\epsilon_t} \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\sigma^2}) \tag{7.5}$$

$$L_x = \log \frac{1}{T} \sum_t \exp(l_{t,c} - \log \sum_{c'} \exp l_{t,c'}) \tag{7.6}$$

7.2 Implementation

As in the binary case, training a model using this loss function with only bounding box labels results in a segmentation of rectangles around objects. Again, a small portion of ground truth labels need to be included in the training set. This way, the model has to minimize both the loss for the ground truth labels as for the bounding box labels. The lowest loss for pixels near objects (background pixels within the bounding boxes) can only be achieved by increasing the uncertainty for bounding box labels.

We keep the same architecture as in figure 6.8. The loss for the ground truth labels and the background pixels for the bounding box labels is now substituted by the cross-entropy loss. The loss for the bounding box pixels is the sum of equation 7.3 and the cross-entropy loss of the flipped label. As we are dealing with multiple classes, we cannot just flip a foreground label to background. Labels are only flipped whenever the uncertainty is high, meaning the model wants to predict a μ for a pixel which is opposite to its label. Assuming the model's prediction is true, the label gets flipped to the model's output based on a threshold for the uncertainty. Due to its sampling nature, when the model wants to lower the loss by increasing the uncertainty, a random set of two classes get high $\boldsymbol{\sigma^2}$. Therefore there only has to be one class with an uncertainty above the threshold, to flip the label.

$$y = \max(\boldsymbol{\sigma^2}) > threshold\ ?\ \text{argmax}(\boldsymbol{\mu})\ :\ y \tag{7.7}$$

7.3 Results

We tested our new method on the cityscapes dataset using the two most frequent "thing" classes: persons and cars. Figures 7.1 - 7.4 show the output and uncertainties of some examples. The figures show the output prediction as well as the "flipping" map for both classes. In the left column, car_bg, car_car and car_person represent the uncertainties for the background, car, and person classes for the pixels labeled car which are above the flipping threshold. The right column represent the uncertainty map for the person pixels. All these pixels are being flipped to the class the network outputs. We notice that most of the erroneous pixels within the bounding box labels are being flipped, resulting in a good segmentation output.

Table 7.1 shows the baseline for 2780 and 500 ground truth images trained with the standard cross-entropy loss. It also shows the result of a model trained with a dataset of 500 ground truth and 2280 bounding box labels. It is clear that training a model with only 500 ground truth images and 2280 bounding box labels can achieve results near the baseline of 2780 ground truth images.

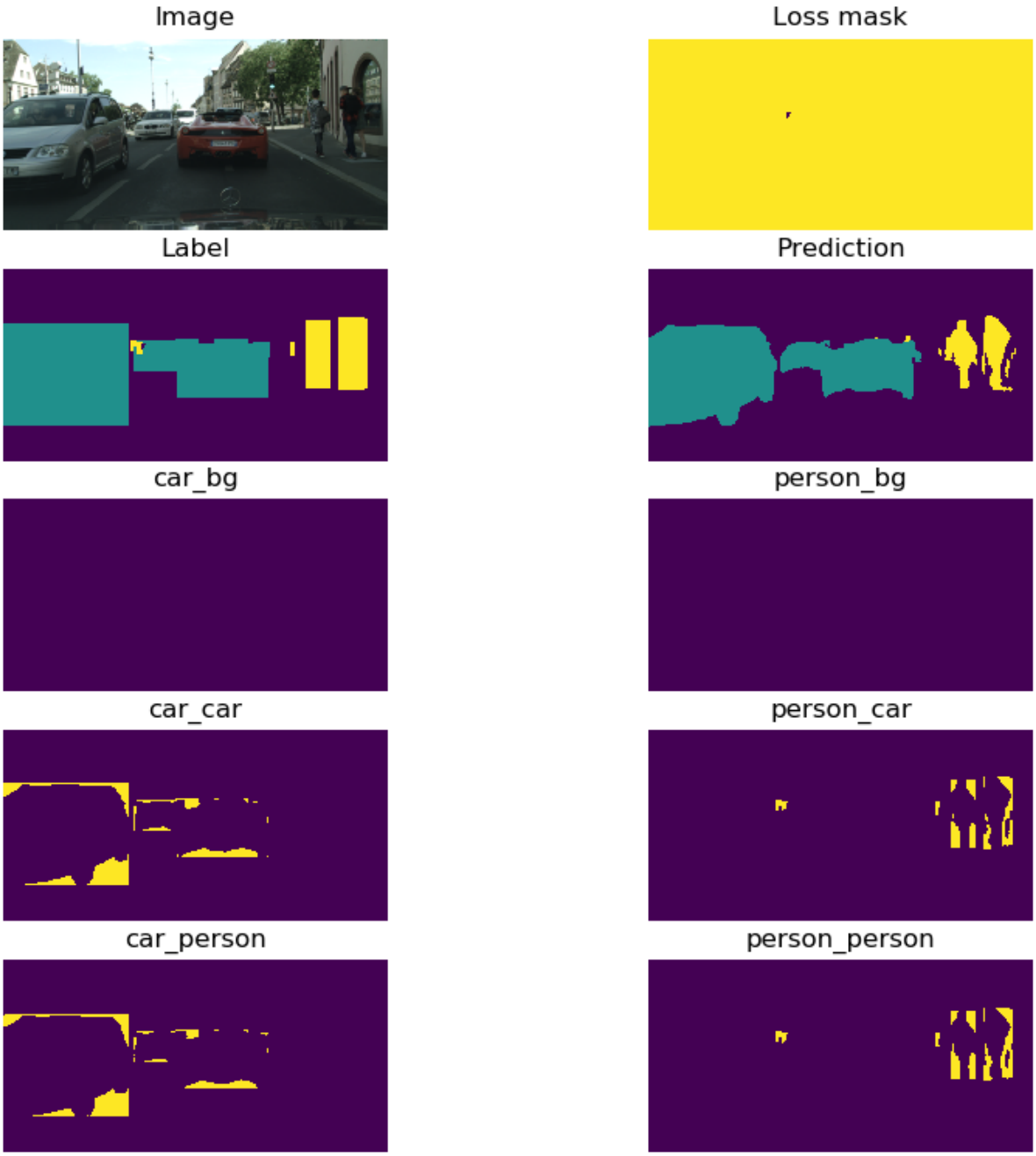


Figure 7.1: Output of the model trained with the loss function described in chapter 7. The last three rows is a binary representation of the uncertainty values for the background, car and person classes. These images display each pixel for which the uncertainty is higher than the threshold value. The uncertainty images shows which pixels are being flipped.

	Dataset		**IoU (%)**			
	GT	***BBox***	***Mean***	***Background***	***Car***	***Person***
Cross-entropy	2780	/	79.80	98.21	85.82	55.68
	500	/	73.36	97.55	79.94	42.58
Our Loss	500	2280	77.75	98.01	82.05	53.2

Table 7.1: The table shows the results for multi-class segmentation. The first two rows show the baseline IoU for training sets comprising of 2780 and 500 ground truth images respectively. The third row shows the result of a model trained with our new loss function on a training set consisting of 500 ground truth and 2280 bounding box images.

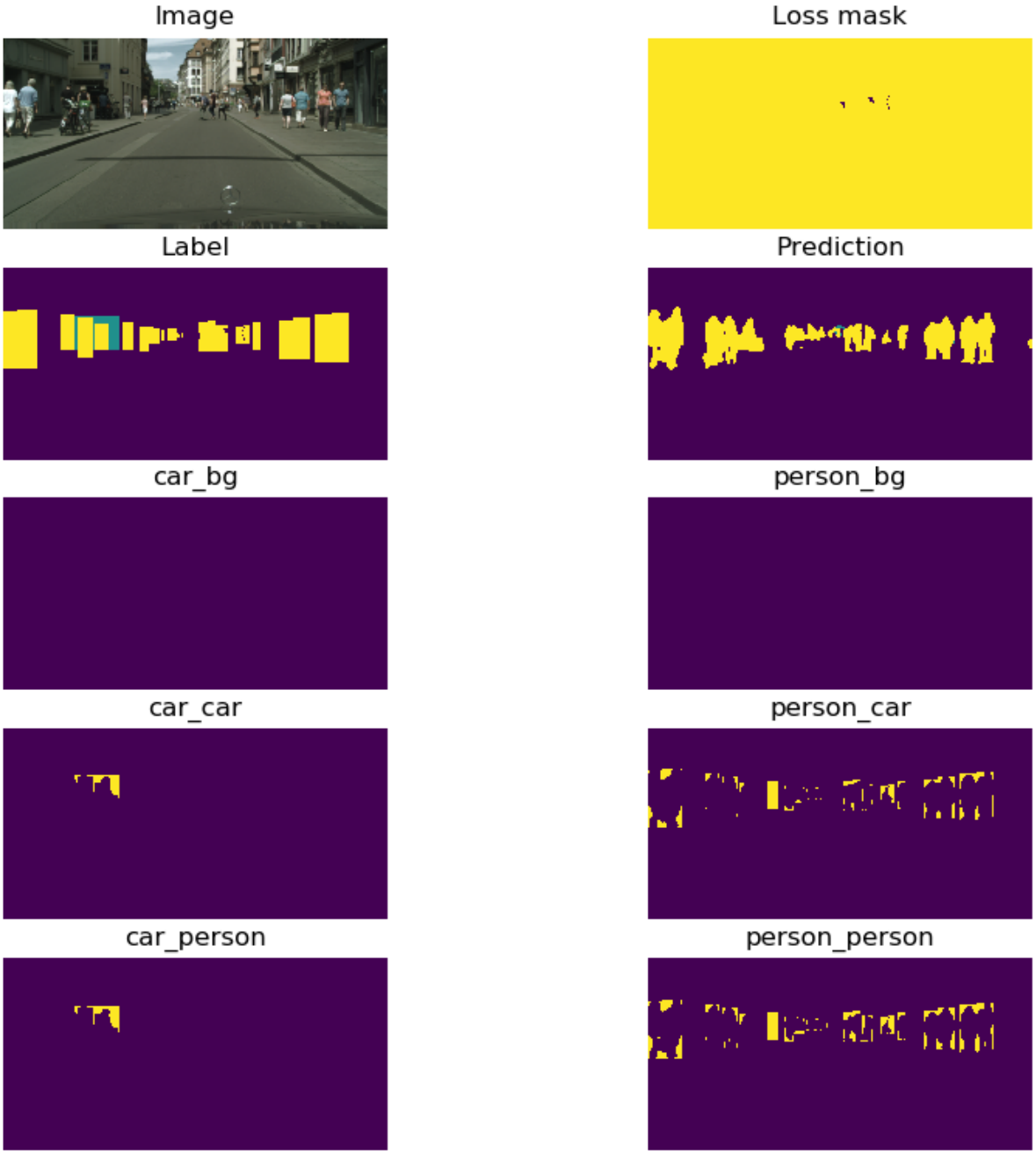


Figure 7.2: Output of the model trained with the loss function described in chapter 7. The last three rows is a binary representation of the uncertainty values for the background, car and person classes. These images display each pixel for which the uncertainty is higher than the threshold value. The uncertainty images shows which pixels are being flipped.

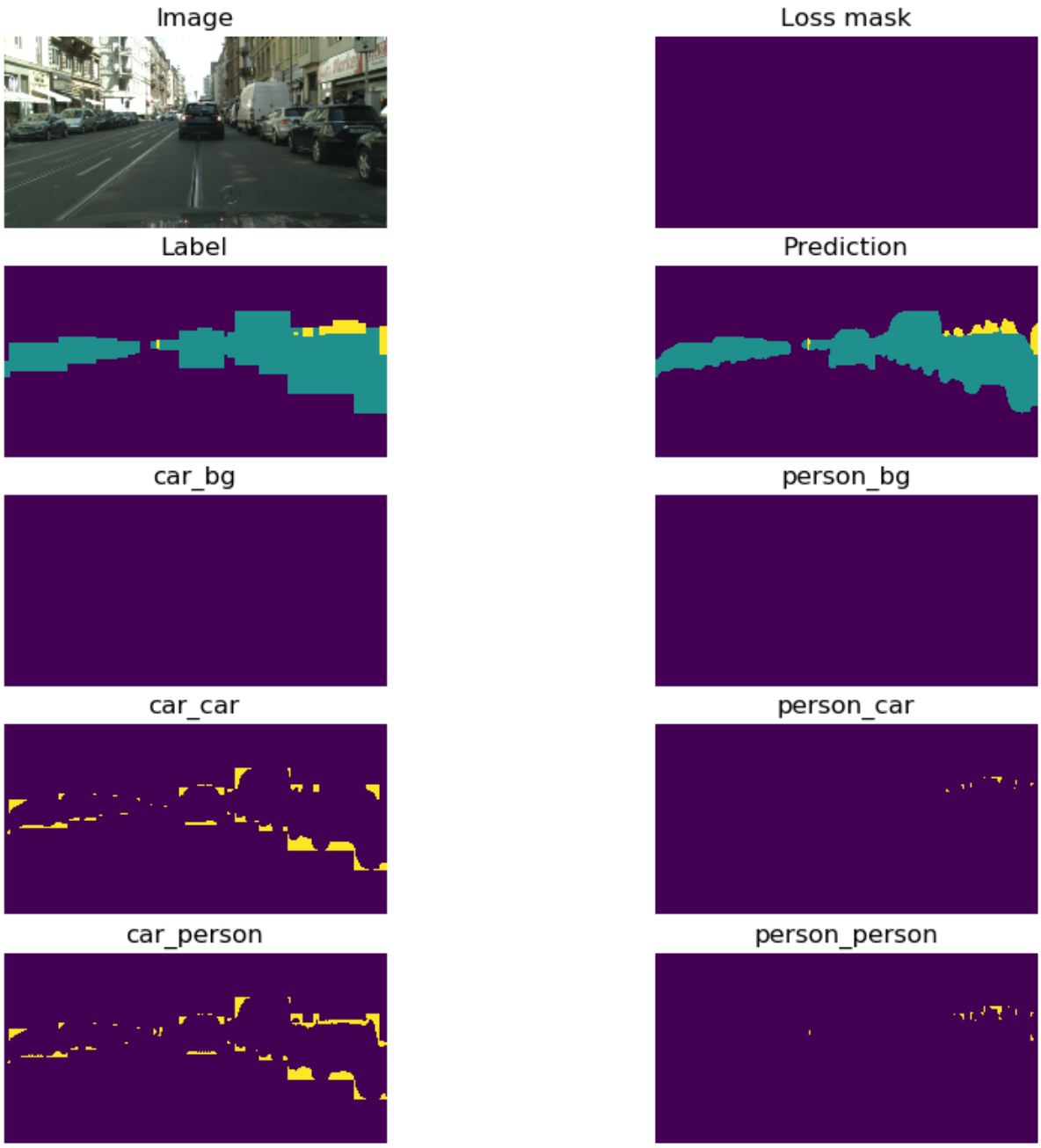


Figure 7.3: Output of the model trained with the loss function described in chapter 7. The last three rows is a binary representation of the uncertainty values for the background, car and person classes. These images display each pixel for which the uncertainty is higher than the threshold value. The uncertainty images shows which pixels are being flipped.

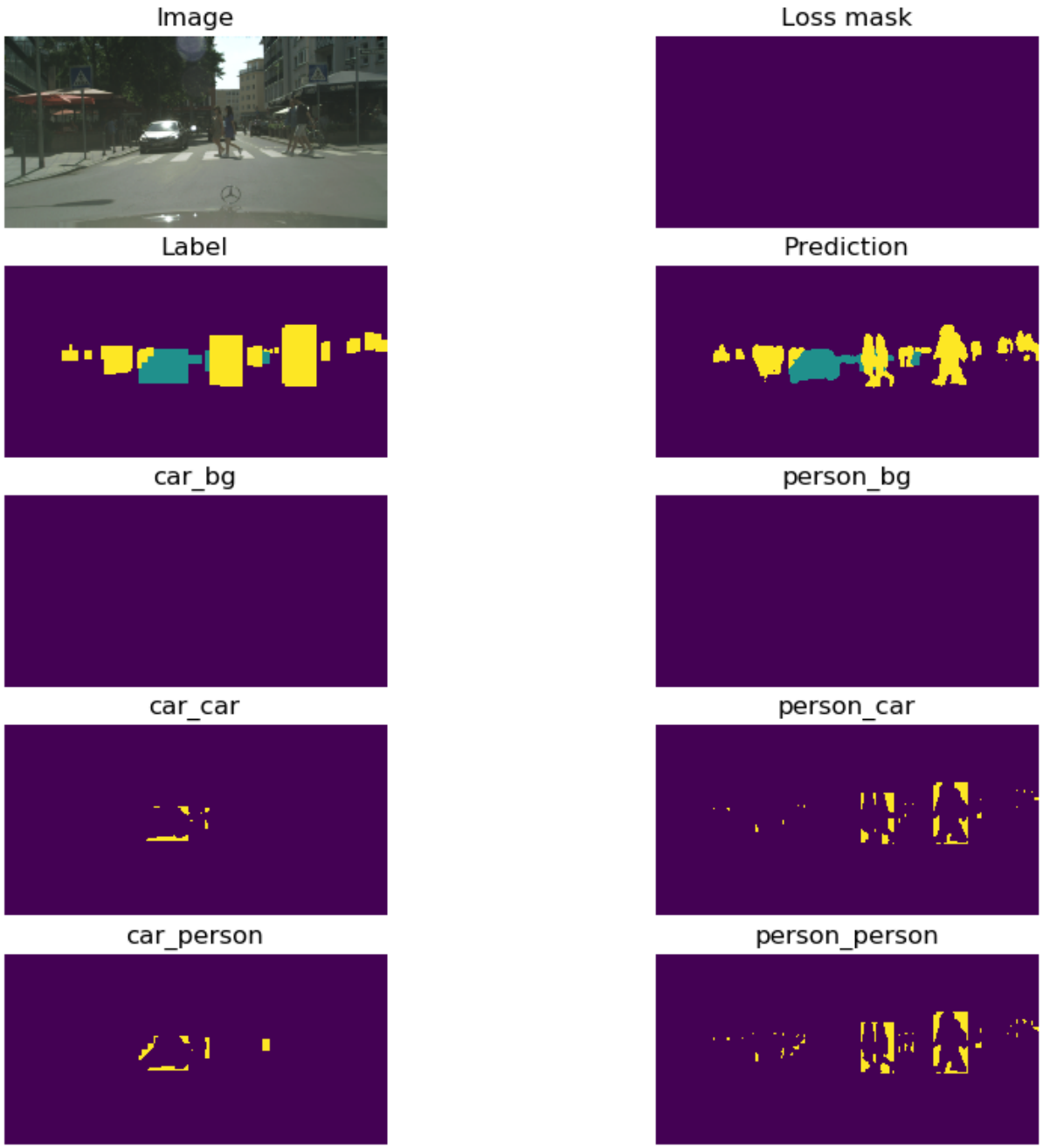


Figure 7.4: Output of the model trained with the loss function described in chapter 7. The last three rows is a binary representation of the uncertainty values for the background, car and person classes. These images display each pixel for which the uncertainty is higher than the threshold value. The uncertainty images shows which pixels are being flipped.

Chapter 8

Conclusion

The objective of this thesis was to train a segmentation model with bounding box labels. In chapter 5, we showed that a regression loss with uncertainty can be used to train a model with bounding box labels only if we include a small portion of ground truth labels. Table 5.1 shows that when limiting the uncertainty only to within the bounding boxes, the addition of bounding box labels results in an IoU increase of around 5%. In chapter 6 we implemented a classification loss function with uncertainty. The biggest drawback of this method were the gradients. The gradients of the loss function with uncertainty forced the model to update the parameters in the direction of the bounding box label. This was fixed by reversing the gradient by flipping the label. By doing so, the labels get updated in function of the uncertainty, resembling a sort of online bootstrapping. Table 6.1 shows that the loss function is capable of getting an IoU almost as high as training a model with only ground truth labels and is around 3% higher than the regression loss with uncertainty. In chapter 7, we expanded the loss function from chapter 6 to a multi-class loss function. The result of this implementation was also an IoU score close to the baseline of using only ground truth images.

This thesis shows a novel way of using uncertainty together with a form of bootstrapping to train a segmentation model on bounding box labels. The only drawback of the method is that it still requires a small portion of ground truth labels. However, since only a small portion of ground truth labels is needed, the increment in IoU of the added bounding box labels is comparable to adding ground truth labels. This means the cost of a dataset can be reduced by having a large part bounding box labels with only a small portion of ground truth labels.

Bibliography

[1] Marius Cordts et al. "The Cityscapes Dataset for Semantic Urban Scene Understanding". In: *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016.

[2] Mark Everingham et al. "The Pascal Visual Object Classes (VOC) Challenge". In: *Int. J. Comput. Vision* 88.2 (June 2010), pp. 303–338. ISSN: 0920-5691. DOI: 10.1007/s11263-009-0275-4. URL: http://dx.doi.org/10.1007/s11263-009-0275-4.

[3] *From MLE/MAP to L2-Loss Regression*. URL: http://shaofanlai.com/post/79.

[4] Alex Kendall and Yarin Gal. "What Uncertainties Do We Need in Bayesian Deep Learning for Computer Vision?" In: *CoRR* abs/1703.04977 (2017). arXiv: 1703.04977. URL: http://arxiv.org/abs/1703.04977.

[5] Diederik Kingma and Jimmy Ba. "Adam: A Method for Stochastic Optimization". In: *International Conference on Learning Representations* (Dec. 2014).

[6] Adam Paszke et al. "Automatic differentiation in PyTorch". In: *NIPS-W*. 2017.

[7] Scott Reed et al. "Training Deep Neural Networks on Noisy Labels with Bootstrapping". In: (Dec. 2014).

[8] Eduardo Romera et al. "Efficient ConvNet for real-time semantic segmentation". In: June 2017, pp. 1789–1794. DOI: 10.1109/IVS.2017.7995966.

[9] E. Romera et al. "ERFNet: Efficient Residual Factorized ConvNet for Real-Time Semantic Segmentation". In: *IEEE Transactions on Intelligent Transportation Systems* 19.1 (Jan. 2018), pp. 263–272. ISSN: 1524-9050. DOI: 10.1109/TITS.2017.2750080.

[10] Li Xiaopeng. *Tricks of Sigmoid Function*. URL: http://eelxpeng.github.io/blog/2017/03/10/Tricks-of-Sigmoid-Function.

[11] Zhi-Hua Zhou. "A brief introduction to weakly supervised learning". In: *National Science Review* 5.1 (Aug. 2017), pp. 44–53. ISSN: 2095-5138. DOI: 10.1093/nsr/nwx106. eprint: http://oup.prod.sis.lan/nsr/article-pdf/5/1/44/24164438/nwx106.pdf. URL: https://doi.org/10.1093/nsr/nwx106.