

Weakly Supervised Roadside Object Segmentation Using Maps

*

J.A.P Guelen

December 1, 2019

*Special thanks to Cyclomedia, my supervisors Albert Ali Salah, Remco Veltkamp, Bas Boom and Julien Vijverberg

Abstract

Publicly available detailed maps offer semantic information of objects in street view images. We investigated if this information can be exploited to automatically create object detection datasets on which (SOTA) state-of-the-art object detection methods can be trained. To accomplish this, we use the Dutch BGT (basis grootschalige topografie) and DTB (digitaal topografisch bestand) maps which are detailed maps of the Netherlands containing the location of a large number of street objects. Using their location information, we find nearby street view images from which we can focus on the position in the image. These images are then collected and labeled with the objects on the map. This allows us to automatically create an image-wide labeled dataset. We also investigated if bounding box and semantic segmentation results could be acquired on this dataset. For this purpose, we used the weakly supervised learning technique of (CAMs) Class activation maps. These CAMs give information about what features in an image lead to a positive classification or negative classification and can give a rough location of where the features are that led to this classification. To improve the accuracy of the CAMs, we investigated two new methods. First, as we used a Resnet 50 network as our SOTA object detection method, we have an additional point from which we could generate CAMs. Normally, when generating CAMs the last convolutional layer, which is a specific layer within a Convolutional neural network (CNN), is used. However, an additional layer within the Resnet 50 network can also be exploited to generate CAMs namely the last 'add' layer of the network. Second, we investigated if the availability of depth information combined with the map information could be used to automatically generate approximate masks of objects that can improve overall performance. We found that our method of automatically generating datasets using maps would generate large numbers of noisy labels. This noisy being present due to mistakes in maps and changes within the environment or due to dynamic obstruction within the road environment i.e. a truck blocking vision of an object. However, even with these faults, we are able to acquire decent image-wide deflections performance acquiring an average F-score of 0.66 on 20 different objects. Where our highest performing object was able to generate a score of 0.92. However, we found our bounding box and semantic segmentation results generated with CAMs very lacking. While an increase of performance could be acquired by using the last 'add' layer for CAM generation this performance increase was not enough to acquire acceptable bounding box and semantic segmentation results. Furthermore, the automatically generated masks were not able to significantly improve results. While an 7% increase in performance at a semantic segmentation level could be seen, this improvement was far from what was required to allow for use able results. Leading us to conclude that while the method shows promise, it requires further research in order to acquire usable bounding box and semantic segmentation results. Our main suggestion for future work is to investigate how the method performs when less noise is present within the dataset.

1 Introduction

Obtaining annotated datasets to train object detection methods is a time consuming process and thus a costly task. In this Thesis, we investigate the possibility of using the semantic information from maps to automatically generate annotated datasets in an effort to reduce this effort.

The problem of detecting and classifying objects in visual media, i.e. images and videos, with computers has been an active research area in the field of computer vision for the last couple of decades and is seen as one of the fundamental goals of this field (2).

The ability to automatically detect and classify objects has led to increased capabilities of automatic systems, allowing them to solve increasingly complex real-world tasks. However, the current state-of-the-art (SOTA) object detection methods learn how to detect and classify objects by using large sets of examples (1) i.e. training sets. This introduces a problem as the creation of these large training sets is exceedingly expensive since obtaining the images and properly defining what can be seen in the image often involves large amounts of manual labor.

Object classification in the field of computer vision refers to the problem of automatically classifying images into a predefined set of classes, where classes are categories of objects with some attribute or property in common. object detection refers to the detection of an object or class in the image while also specifying the location of that object. Recent object detection techniques have seen increased performance as new methods were developed. This increased performance has led to useful real-world applications that previously were not possible. An example is the creation of algorithms that can automatically detect tumors in people (42).

Another recent possibility opened up by the ever-increasing performance of object detectors is that autonomous driving is now within the realm of real-world use. Autonomous driving is the idea of allowing vehicles to drive and control themselves partially or completely automatically. The current SOTA autonomous cars can drive themselves successfully in specific well-defined environments. For example, highways were one of the first locations where autonomous driving was possible as they have clear and simple rules with few difficult maneuvers. But they still struggle in locations that have context-dependent rules, where exceptions are prominent and in yet unseen and untrained situations that require specific answers (59; 60).

Accurate object classification and detection of all objects in the road environment is crucial for understanding more of the context of a certain location. Currently, the focus is on detecting objects that directly impact driving, e.g. road markings, other vehicles, traffic signs. Having the ability to detect more of the objects within the road environment could help improve autonomous driving, as knowing what other objects are present in the current environment could help autonomous cars give context to certain events. For example, by knowing where street lights are located, an autonomous car could determine where it is on the road, even when road markings are faded or not visible.

Learning all these new possible objects is a challenging task but can be accomplished by using SOTA methods. These methods, however, require large training sets/datasets which become increasingly expensive as more object types are added. Furthermore, within the roadside environment, there can be a high amount of inter-class variation as shown by Figure 1. This inter-class variation is increased when going over country and region boundaries as there is not a standardized appearance for many road-side objects. This amount of inter-class variance may require country/region-specific

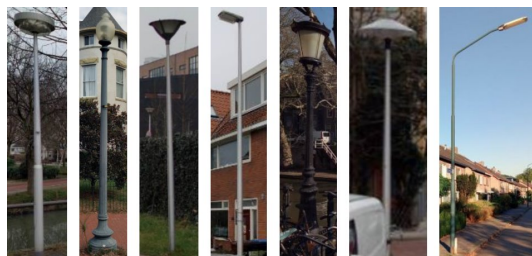


Figure 1: Example of the variance within the class street light.

models and training sets which would further increase the cost of creating accurate detection models that are capable enough to allow for autonomous driving in multiple countries.

Furthermore, there are also different types of labels, with different cost associated with them, that give different information about objects in images. The most basic type of label might only give image-wide information, e.g. "There is a **cat** in this image", without specifying where the particular object actually is. The next level of labeling produces a square rectangular shape, called a bounding box, which is drawn around the object of interest (2; 18) e.g Figure 2. Methods that produces bounding boxes are known as object detection methods. While bounding boxes give a rough estimation where the object is and what size the object has within an image, when a more precise shape of the object is necessary segmentation is used. Segmentation segments the image on a pixel by pixel basis i.e. for every pixel we determine if it is part of the object/class. These pixels then together create a mask of the object/class.

A further distinction can be made within this segmentation: Semantic segmentation and instance segmentation. In semantic segmentation, segmentation is done on a class level e.g. all cows in an image fall under the class cow and are segmented as a single mask. In instance segmentation, segmentation is done on an instance level e.g. all cows in an image have their own segmentation and their own masks. The difference between these forms of labeling is illustrated in Figure 2. While these more information-rich forms of labeling will lead to more rich forms of predictions, they are also often far more costly to create.

These harder kind of predictions do allows for additional application that without them would not be possible. For instance, object avoidance requires knowledge on where an object is located an image-wide detection in those cases would not be sufficient. Furthermore, when wanting to analyze faults or mistakes in an object, being able to create a mask that perfectly encapsulated the object would allow further methods to focus on detecting faults within the generated the masks. This would simplify the following models significantly as they do not have to deal with the process of finding and recognition the object.

Fortunately, there are methods to reduce the cost of creating new datasets, an almost universally used method within computer-vision is to use pre-training. Pre-training - a form of transfer learning - in object detection refers to the ability to get increased performance by first training on a large existing dataset, for a different task, before training on a more specific classification task (6). This can significantly reduce the size required to acquire high quality high level labels. Two popular large existing datasets used for this are ImageNet (44) and Microsoft's CoCo (43). By learning

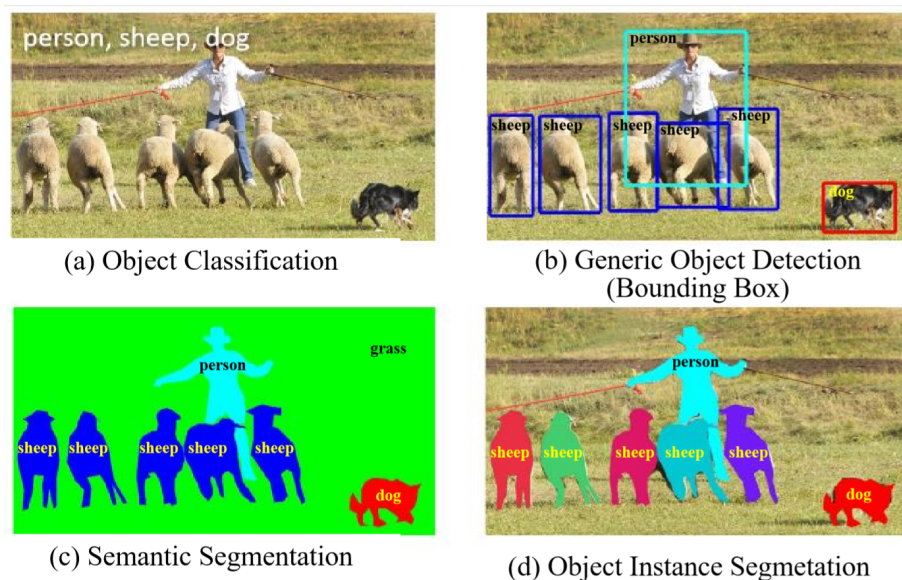


Figure 2: Different types of labels, image from Liu et al. (2).

on these datasets first the method can already learn common interchangeable patterns within the images. This reduces the number of images needed for training as classification knowledge can be transferred for new objects more easily compared to relearning them from scratch. This is a prominent first step in the learning process of current SOTA methods (6). However, even with transfer learning, creating a training set with semantic or instance segmentation labels is still very time consuming and considered a costly undertaking. Another recent method that can significantly reduce the cost of creating these datasets without lowering the type of label predictions is known as Weakly Supervised Learning (WSL).

WSL in computer vision refers to the task of learning on a training set where labels are less precise than the desired output also known as weaker labels as illustrated in Figure 3. WSL has the advantage that it requires less strict forms of labeling. This allows for the use of simpler forms of labels, for harder problems. More importantly, it allows for image-wide labels to be used for semantic segmentation. The creation of image-wide labeled datasets is considerably easier and in some cases the creation of such a dataset can be automated. This can drastically reduce the cost of creating such a dataset. A downside of this method, however, is that the information missing from the labels has to be acquired via different means as information lost by this weaker label, like location information, has to be acquired via some different way.

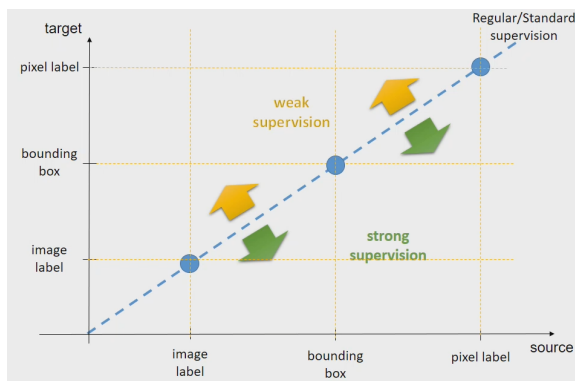


Figure 3: The different types of learning image from Bilen (54).

1.1 Goals And Challenges

In this Thesis, we investigate the possibility of using semantic information from maps to automatically obtain image-wide labeling for static roadside objects. The key idea is that if an object is labeled on a map, we can use its location to find a street view image nearby and focus on that object. Thereby creating an image which we could now label as containing that object. Repeating this enough times for enough objects, we hypothesize that this can then be used as an effective training set for learning object detection on static roadside objects. If successful this would drastically reduce the cost of creating large training sets for the task of object detection of static roadside objects, as this method can almost completely automatically generate such a training set. However, the method will be largely reliant on the quality and accuracy of the map labels, and the noise in the street environment. Figure 4 and 5 illustrate two ways how label noise can be introduced even when an area was perfectly and accurately labeled. First, dynamic objects situated in the street environment can introduce noise as they can block direct sight on the object. Second, noise can be also be introduced due to the difference in time between the map measurement and the image capture time. It is therefore key that the maps and street view images are as accurate in terms of gps locations and objects being correctly labeled as possible, as the method itself can introduce large forms of noise. In this research, we investigate if the public dutch BGT and DTB maps are sufficient in fulfilling that role for maps. For street view images a private dataset from Cyclomedia is used which contains roadside images across almost all areas covered by the BGT and DTB maps (see Section 3).

Our first goal is thus to devise a method that can automatically construct a dataset from map information and street view images, (see section 4.1), and to investigate the quality of this dataset (see section 5.2).

Our second goal is to investigate what performance can be achieved on the automatically generated dataset. This performance will be measured on three levels: Image-wide classification, bounding box generation and semantic segmentation. However as our only training is on image-wide labels, WSL will have to be used to acquire bounding box and semantic segmentation results. Acquiring result on these harder forms of labeling will be a challenging task as we have no direct location information available. While methods exists that can acquire this information from image-wide



Figure 4: Visibility of a map labeled object blocked by a dynamic object.



(a) Image taken in 2015



(b) Image taken in 2018

Figure 5: Two images taken at different times of the same location that should contain a bench according to map labels. Unfortunately, the label is no longer accurate due to changes in the environment.

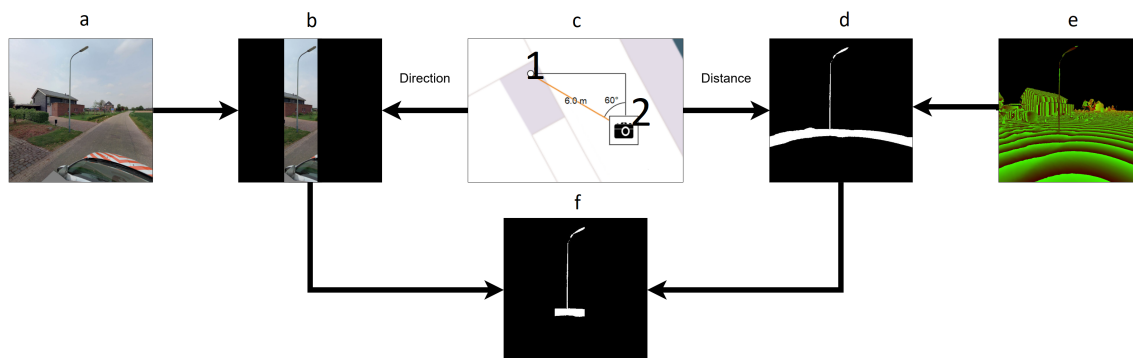


Figure 6: Example of how map information plus depth information can be combined to help localize the object. Where (a,b) show how a direction is used to filter the object location from the image, (c) shows how the direction and distance are calculated between (1)(2) which are the location of the object and image respectively. Then, (d,e) show how a range on the distance is used to filter out background and foreground. Finally, (f) shows the automatically generated rough mask when (b,d) are combined.

defections, we will also investigate using depth information combined with map information.

As we have location information from the maps and a position of where the image was recorded, we can calculate the distance and rough direction to the object within an image. If we use this information with a depth image, it allows us to filter out everything before and after a certain distance. When combining this with an approximate range on the direction, we are able to generate a rough mask as shown in Figure 6. This mask can then be used during training to help give us location information and help guide or WSL method. See section 4.3 for more detail.

1.2 Research Questions

These two main goals lead us to the following research questions.

1. Can the information on maps, combined with street view images, be used to create a dataset capable of training object detection methods?
2. What performance can be achieved on this automatically generated dataset on an image-wide, bounding box and semantic segmentation level?
3. Can location information acquired by leveraging depth information combined with map information improve bounding box and semantic segmentation performance?

1.3 CycloMedia

This thesis was created at CycloMedia¹, a company specializing in providing street-level imagery with high accuracy. To create the street-level imagery, they use an advanced five-camera panoramic image capture system. This system was developed in house and is put on top of cars that drive around, taking high-quality images while also creating depth images by using lidar. They use these images and depth images to populate their cloud platform, which allows customers to easily view public locations in high quality. They also annotate these images on customers request with highly detailed information. CycloMedia is based in the Netherlands and has customers around the world.

CycloMedia's interest in this research is for reducing the cost of needing to manually annotate street view objects while they are already present in maps. Furthermore, the automatic creation of new, highly detailed maps could be a promising feature to sell to customers.

2 Related Work

In this section will go over related work, starting off by describing the SOTA with regards to object detection methods. This is followed by unsupervised learning methods that also reduce the cost of dataset creation. Finally, we investigate currently popular weakly supervised learning methods.

2.1 Object Detection In Computer Vision

object detection has seen great improvements in recent years, this is mainly a consequence of the increased computing power enabling the advancement of new techniques. One of these techniques in recent years was using Deep Convolution Neural Network (CNNs) for image classification as shown by Krizhevsky et al. (1). This work showed a huge performance increase over existing methods and showed that CNNs could get high performance in object detection. However, this new method also introduced new challenges.

2.1.1 Neural Networks

CNNs are a type of neural network, where a neural network (NN) is a network of neurons connected to each other in a particular way. These neurons are nonlinear functions that produce a sequence of real-valued activations based on their inputs and some bias (17).

While some neurons have inputs directly from an input neuron, most have inputs through weighted connections from other neurons. In a neural network, neurons are usually structured in layers where neurons in one layer are connected to the neurons in another layers, illustrated in Figure 7. When the neurons in one layer are fully connected to the neurons in an next layer, i.e. every neuron in one layer has a connection to all the neurons in the next layer, we call this a fully connected layer.

Once a neuron gets inputs, it performs a computation and then passes its result to the next neuron that is connected to it. In every connection between two neuron a weight is present which multiply

¹ for more information about CycloMedia see <https://www.cyclomedia.com/en>

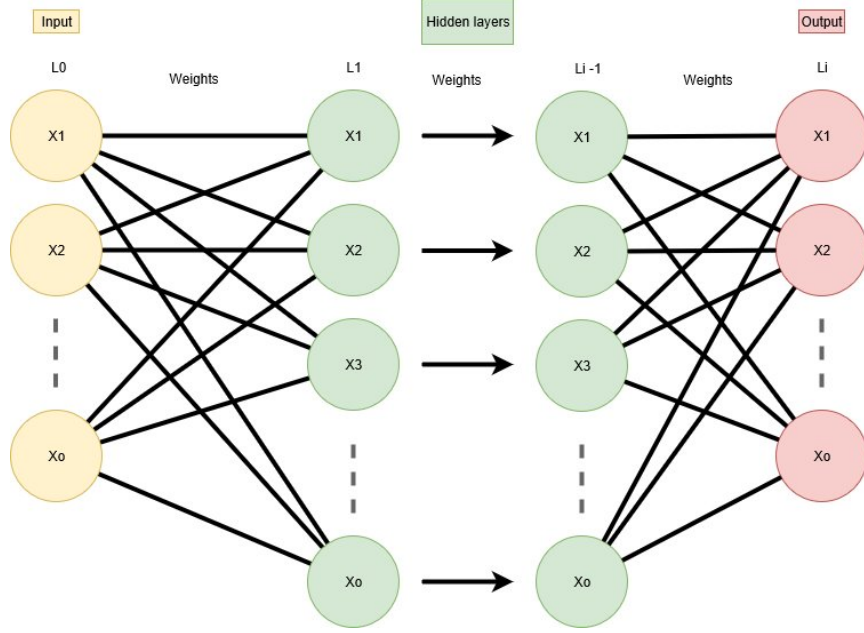


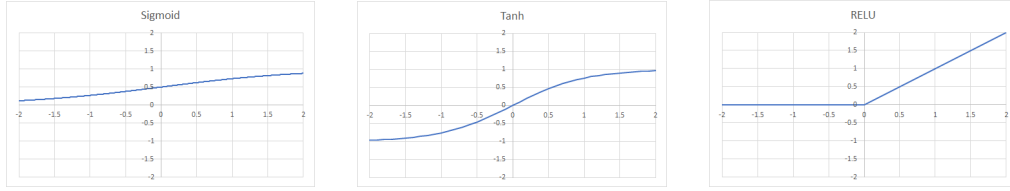
Figure 7: A neural network consisting out of multiple layers with multiple neurons arranged in a feed forward structure.

the output value of a neuron. The inputs in a neuron are then usually summed after which an the activation function denoted as ϕ performs a nonlinear computation. This leads to neuron X_o in layer L_i giving output y_{io} with given inputs $X_n \in L_{i-1}$ where n is the number neurons in layer L_{i-1} . The weights between neuron X_n in layer L_{i-1} and neuron X_o in layer L_i we denoted by w_{ion} . A bias is often added to each neuron as it allows the neuron the be more finely tuned and add a certain value during computation instead of only modifying input. To easily add a bias to the network, the number of input of each neurons is increased by one and we place an additional neuron with output one at the beginning of this list of inputs. The bias can then be changed and learned by changing the weight of this extra neuron, i.e. $x_0 = 1$ and $w_{io0} = bias$, resulting in

$$y_{io} = \phi\left(\sum_{k=0}^n w_{iko} X_k\right). \quad (1)$$

Several different activation functions exist with the main goals of an activation function being to map the resulting values from input into a certain range, e.g. ≥ 0 , 0 to 1 or -1 to 1, and to add non linearity to improve classifier performance. This kind of remapping allows for non-linear characteristics to be introduced into the network. As without such remapping the node activation would be limited to linear combinations of input (64). Commonly used function include the Relu (66), Sigmoid functions (65) and Tanh function. The Sigmoid, also knows as logistic, activation function is structured in such a way that its output is always between 0 and 1, i.e.

$$\omega(x) = \frac{1}{1 + e^{-x}}, \quad (2)$$



(a) Sigmoid Activation function (b) Tanh Activation function (c) RELU Activation function

Figure 8: Example of the different activation functions

and its output is illustrated in Figure 8a. The Sigmoid activation function has the advantage that it has clear bounds for the activation function. Furthermore, the Sigmoid activation function balances values closer to 0 or 1 making the neuron more discriminatory towards inputs. The Tanh activation function has a shape similar to the Sigmoid function but is structured to return -1 to 1, i.e.

$$\omega(x) = \frac{(e^x - e^{-x})}{(e^x + e^{-x})}, \tag{3}$$

resulting in the output as illustrated in Figure 8b.

This makes the function even more discriminative and allows for a clear distinction to be made between a positive activation +1 and a negative activation -1.

The Relu activation function simply outputs the sum of the input as long as it is greater or equal to zero else it outputs zero, which results in an output between 0 and ∞ i.e.

$$\omega(x) = \begin{cases} if & x \geq 0 : & : x \\ o/w : & 0 \end{cases}, \tag{4}$$

resulting in Figure 8c.

This Relu function, while much simpler than Tanh and the Sigmoid activation functions, has a distinct advantage that in a large network a few neurons could lead to a strong estimation as these neurons are not limited to a max output of 1. This reduces the effect of the vanishing gradient problem, which is the problem where as the magnitude of the impact of a neuron get smaller and smaller the bigger the overall network is, the impacts get to small for any impact full changes to acquire. Using Relu can reduce the problems caused by the vanishing gradient problem as less of the network has to activate for a given classification, thereby, increasing the magnitude of those that do. However, this also introduces a problem as only a few neurons respond to a classification this means the other neurons do not participate in training resulting in them also not being adjusted. This is also known as the dying Relu problem (73). Choosing an activation function thereby impacts the time the network needs to converge.

Once an activation function has been chosen and a network of neurons has been constructed it needs to be trained, in order to learn a function $f(x) = y$ where y should be as close to the ground truth. This is accomplished with a process called back-propagation.

The first step of this process is to randomize all the weights and biases within the neural network. This is done to break up the symmetry within layers as if two weights leading to a single neuron are equal they will also always remain equal to each other, a downside of using back-propagation (67).

Back-propagation tries to minimize the error of the network. It does this by changing the weight between neurons and tries to find a direction in which to move each weight, with the goal of reducing the incorrectness of the network. This measure of incorrectness is known as the loss and this loss is calculated usually over a batch of examples which is a subset of all examples ². This loss is calculated by first letting the neural network make a prediction on a given input. A function then compares the output of the network, i.e. predictions, with the expected result. The function that calculates the difference between the two is called the loss function.

Many different loss functions exist all with their own benefits and downsides. A commonly used loss function for classification is the square loss function. In the square loss function the loss is calculated by taking the difference between the real label $\hat{\theta}$ and the prediction label θ and squaring this difference. Resulting in $Loss(\hat{\theta}, \theta) = [\hat{\theta} - \theta]^2$. For back-propagation, the loss is usually multiplied by 0.5, resulting in $Loss(\hat{\theta}, \theta) = 0.5[\hat{\theta} - \theta]^2$, this is done to make calculating the derivative easier, which is needed for the back-propagation. One issue with this loss function is that because of the squared part, it reacts very strongly to outliers i.e. outliers can have a large impact on the loss function, which in turn can greatly effect the back-propagation algorithm. A simple solution for this is to use the absolute loss function where instead of squaring $\hat{\theta} - \theta$ we take its absolute value, resulting in the function $Loss(\hat{\theta}, \theta) = |\hat{\theta} - \theta|$. However, this makes the function a non-differentiable function which makes calculating the derivative a lot harder. After averaging the loss of all examples in the current batch, back-propagation calculates how much every weight needs to adapt to do better in the current training batch.

Gradient descent is used to try to move in the direction where the loss is minimal. To achieve this, one needs to calculate the derivative of the loss in regards to how each weight and bias impacts this loss i.e. how does every change to one weight or bias effect the output of the network. To calculate the loss of a certain weight w in layer i that connects neurons n with neuron o , where n is the neuron in the previous layer and o the neuron in the current layer one needs to know what impact this has on the output of the network.

To find how one weights impacts the outputs of the network, we need to know how the neurons and weights after it impact the output of the weight i.e. we need to know how the weights and neurons after it effect its output. We, therefore, start at the end of the network and propagate the effect backwards through the network i.e. backpropagation. This results in

$$\frac{\partial E}{\partial w_{ino}}. \tag{5}$$

Where E is the error of the network. This equals to

$$\frac{\partial E}{\partial w_{ino}} = \frac{\partial E}{\partial y_{io}} \frac{\partial y_{io}}{\partial net_{io}} \frac{\partial net_{io}}{\partial w_{ino}} \tag{6}$$

Where $\frac{\partial E}{\partial y_{io}}$ is the impact of the output of the neuron o in layer i . So starting at an output neuron o $\frac{\partial E}{\partial y_{io}} = loss(\hat{\theta}, y_{io})$, the loss is propagated backwards to previous layers. To accomplish this it first needs to be propagated back through the neurons activation function $\frac{\partial y_{io}}{\partial net_{io}}$ after which this needs to be propagated to all the inputs of the neuron $\frac{\partial net_{io}}{\partial w_{ino}}$ as illustrated in Figure 9. This process is

²This is also known as minibatch learning.

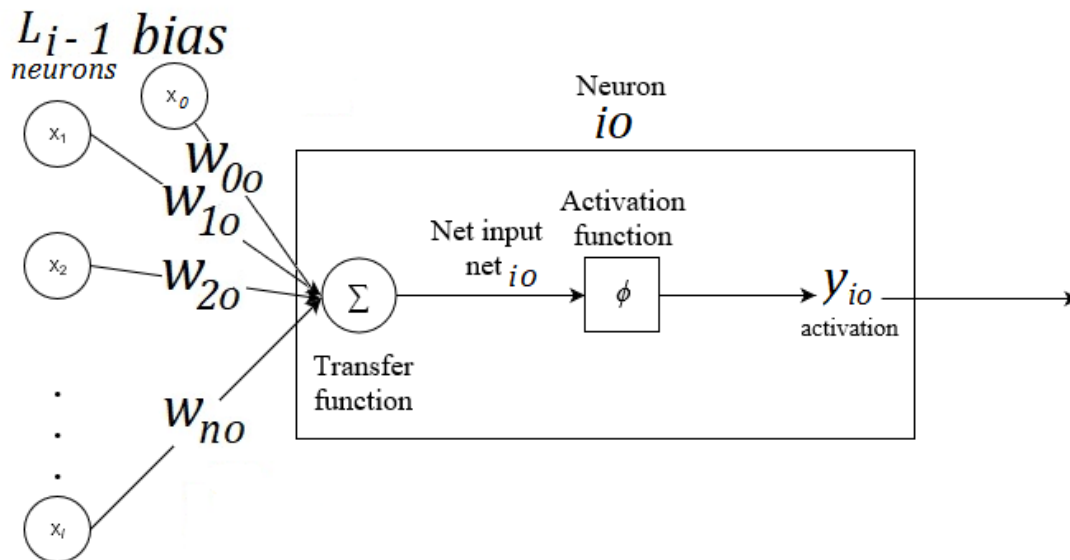


Figure 9: Example neuron j , Modified from Chrislb (68).

repeated for all output neurons resulting in a list of impacts for each weight and bias. As these impacts are calculated on the derivatives of the loss, they give a direction in which to change the weights or bias of a neuron. By summing this list of impacts per weight or bias we get one direction to move the weight or bias to.

This direction is then multiplied against a learning rate parameter. This learning rate parameter (usually situated within $[0, 1]$) dictates how fast the function converges. However, setting this parameter too high might result in overshooting while setting it to low would increase training time. Therefore, this learning rate usually starts high and lowers the longer the training has been going on.

Sometimes a weighted loss is used that weighs the loss for the different classes differently. This class weighting is usually added after the loss function and reduces the impact of a certain class. This can be desirable when there are class imbalances within a training set which can result in certain classes not being able to be learned as the loss is mostly focused on a larger class. By incorporating smaller weights for these over-represented classes compared to classes that are underrepresented it allows the network to learn both classes.

2.1.2 Convolution Neural Network

A Convolution Neural Network (CNN) is a type of neural network with specific special layers. One of the more important special layers is called a "convolutional layer", which is a layer where

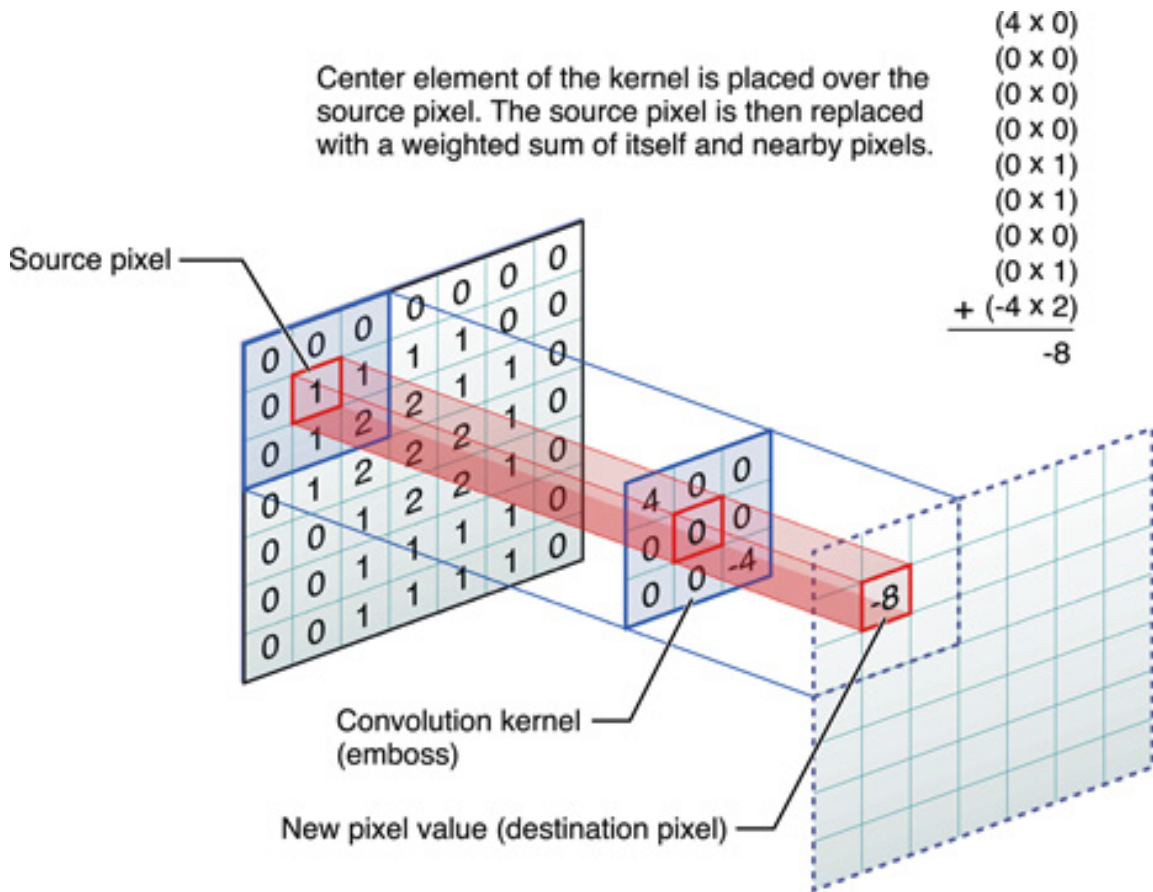


Figure 10: Convolution layer, image from Apple (62).

typically a rectangular filter is shifted over a N-dimensional array of inputs. This filter has certain activation events as illustrated in Figure 10.

The intuition behind this layer is simple, the convolutional layer allows for the detection of low-level features e.g. a straight line, a curved line. These fundamental features can then be used in new convolutional layers allowing the detection of higher-level features e.g. a combination of lines creating a simple shape. Multiple different filters are used in each convolutional layer, where each filter is an extra dimension added to the original 2d dimensional array, creating a 3d feature space.

Another commonly used layer in CNNs is called a pooling layer. In a pooling layer, the inputs over a certain area are pooled together according to predetermined criteria. For instance, max-pooling takes the maximum value out of some region, while average and minimum pooling take the average and minimum values, respectively. These pooling layers are used to limit the size of CNNs and also help in finding location and rotation invariant features (41).

A CNN is constructed out of multiple convolution layers combined with pooling layers in a feed-

forward structure³. The final layer is then a dense Fully Connected (FC) layer with a size equal to the number of labels. This final layer has the goal of providing an output on which the ground truth labels can match i.e. an array of labels and scores indicating if an object is present within an image.

This final layer also has a few different variants. For instance, there is the commonly used softmax layer, which normalizes the inputs on a probability scale where the highest value of the input is over-represented within the probability scale. It is commonly used as it is simplistic and provides a probabilistic interpretation (69).

Another commonly used last layer is a Sigmoid layer. This is commonly used instead of softmax in multi label problems, which are problems where multiple labels could be needed to correctly describe the input.

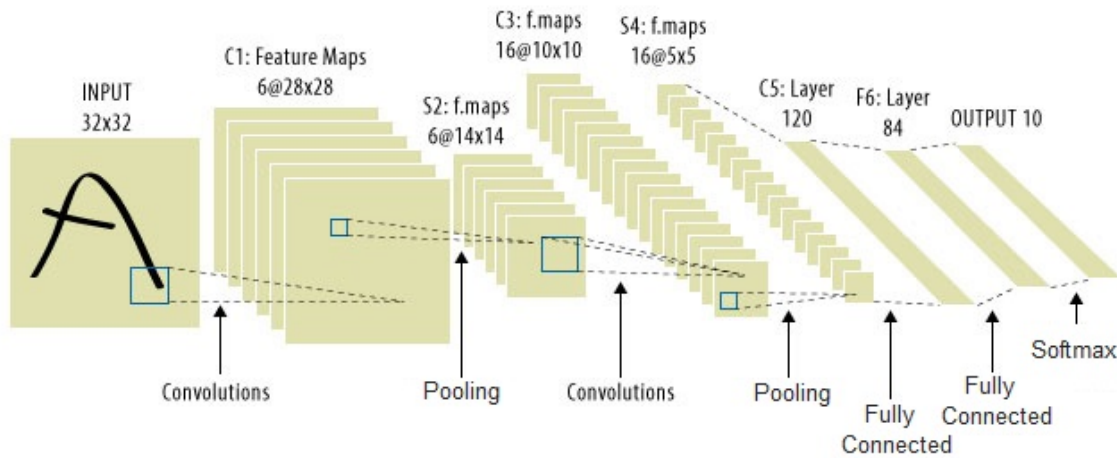


Figure 11: An example of the structure of a CNN, image modified from LeCun et al.(63).

CNNs are also deep learning as they consist of many layers (17). Deep learning helps the network with classification, as it allows learning distinct features. However, it also introduces a problem as increasing the number of layers also increases the number of weights and biases of needing to be accurately set. Thereby requiring a larger dataset to correctly train these extra weights. Furthermore, this increased size increases the training time, requires more computing power and needs more memory to be able to represent the network.

2.1.3 Object Detection CNNs

Krizhevsky's et al. (1) showed that CNNs could achieve better results for object-classification task than other (at the time) SOTA methods on a large amount of classes. They showed this by creating a CNN network called Alexnet with which they won the ImageNet LSVRC-2010 contest (72). In

³ A feed-forward network is a network without any loops or circular flows.

their paper, they showed numerous methods of increasing the performance and capability of CNNs. They showed that a CNN can be trained over multiple GPU effectively allowing for larger sized networks as more memory is available. Furthermore, they showed that they could reduce overfitting by dropout layers. These layers, dropout input with a certain chance (50% in Alexnet). These dropouts would then not be taken into account when the network would forward pass or backward propagate. Krizhevsky's et al. found that these layers could reduce overfitting at the cost of time for convergence. Krizhevsky's et al. also showed that the depth of the CNNs was really important and removing layers would lead to performance decreases.

Girshick et al. improved on Krizhevsky's et al. CNN method with a new method called R-CNN (6). In R-CNN they introduced using a region proposal algorithm to segment the possible objects before inputting them into a CNN. Region proposal algorithms are algorithms that provide suggestions for dividing the image into sections (70).

Previously, only sliding window approaches were used for CNNs to accomplish this. This sliding window approach requires an estimated object size, which then becomes the size of the sliding window. This window then slides over the image with an predefined step size, and every window step is then a new region. Depending on image size and step size, this could generate a large number of regions. One problem is that the sliding window size is fixed, while the objects size changes depending on the distance to the camera. Furthermore, the aspect ratio is also fixed making it unable to deal with rotated objects.

Therefore Girshick et al. instead chose to use selective search as introduced by Uijlings Et al. (7). Selective search works by first over-segmenting the image based on color, texture, size, and shape. Working from this over-segmented image, areas adjacent to each other are combined based on similarity and placed into a new layer. In this new layer, the process is repeated again and new areas can be combined again with other adjacent areas. This process can be repeated until the whole image is one region, see Figure 12. The advantage of this is that it creates multiple levels and layers of segmentation, which can then be selected by certain criteria e.g. in the case where you want a lot of smaller region proposals then you would include more of the lower layers. Finally, these proposals can be turned into bounding boxes by putting the closest fitting box around each region proposal.

This region proposal algorithm allowed for the uninteresting parts of the images to be disregarded and focused to be placed on the interesting parts of an image, thereby significantly increasing performance in both speed and accuracy of the CNN. They also found that transfer learning can be highly effective for a CNN i.e. training on generic images followed by domain-specific tuning yielded a significant performance boost (6).

After the publication of R-CNN multiple improvements where suggested. He et al. proposed a method called SPPNet (8). SPPNet removed the fixed images size requirement that was needed for R-CNN and CNNs, as fully connected layers have a fixed predetermined size. They achieved this by introducing a new layer to CNNs called a "spatial pyramid pooling layer" (SPP).

The spatial pyramid pooling layer pools the image by bins that have certain sized areas. The first pyramid pooling bin pools an area equal to the size of the entire image. The following pyramid pooling bins pool in increasingly smaller areas of the feature space. These bins are then combined into a fixed-size representation needed for the fully connected layers as can be seen in Figure 13.

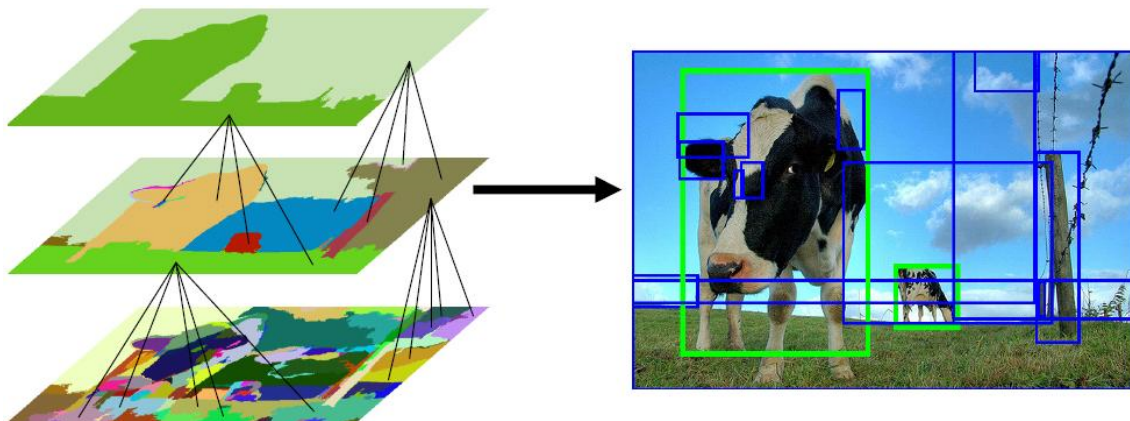


Figure 12: Selective search, image from Uijlings et al. (7).

The SPP layer gives the advantages that data preparation steps and data augmentation steps, by resizing, become much simpler as no requirements are placed on the image size. Furthermore, in cases where resizing was mandatory, warping and cropping could lead to a decrease in overall performance, which can now be avoided. They also increased the speed of running the resulting SPPNet on test images by only calculating the convolutional features of the whole image once as opposed to R-CNN where these features are calculated for each region proposal separately.

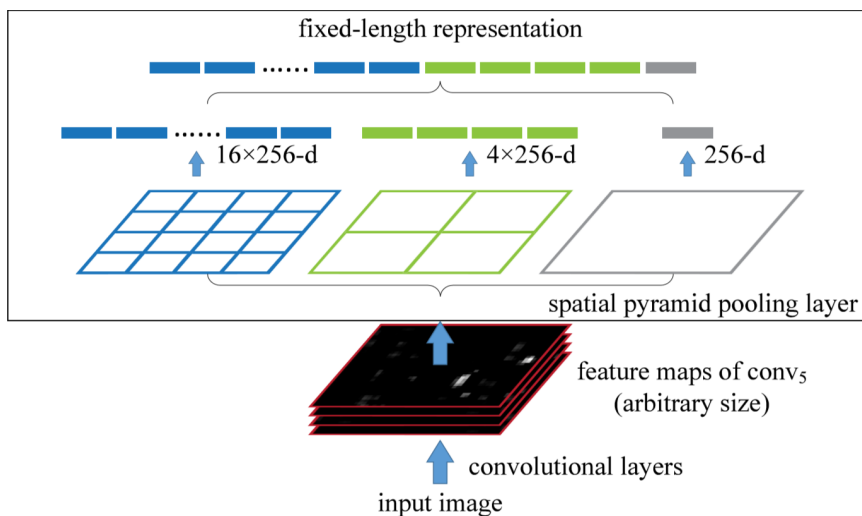


Figure 13: Spatial pyramid pooling layer image from Kaiming He et al. (8).

SPPnet was followed by Fast R-CNN (9). Fast R-CNN sought to increase the performance of SPPNet and R-CNN. This was done by using region proposal not on the image itself, but on the feature space created by convolutional layers. This technique in a sense shared resources of the

R-CNN with the region proposal algorithm, thereby increasing performance. They also proposed adding a region proposal pooling, i.e. region of interest (ROI) pooling layer, which sought to do what SPPnet also did, allowing for variant image size, but only for region proposals. They achieved this not by using a full pyramid pooling layer, but instead by only using one layer of the pyramid pooling layer, where they could change the resulting size of the image to any size. For example, if they wanted to get 7x7 images, they could create an ROI pooling layer that would max pool the images to a 7x7 image. This new ROI pooling layer led to increased performance as they found that this single scale layering had a performance almost as good as a pyramid pooling layer with significantly reduced cost.

With these new methods, the overall performance of CNNs increased and the added benefits of pre-training and the ability of using differently sized images increased overall flexibility. Recently, focus shifted to increasing the size of networks as it was found that the deeper the network the better performance could be acquired. Furthermore, focus shifted to more detailed forms of detection, namely semantic and instance segmentation.

2.1.4 Deep CNNs And Semantic And Instance Segmentation

To allow for semantic segmentation Long et al. (53) introduced FCN, fully convolutional networks for semantic segmentation. In fully convolutional networks, the networks only consist of convolutional layers and pooling layers. This removes the requirement of fixed-sized images needed for the fully connected layers as fully connected layer have a fixed size requirement which is not present within an FCN. Furthermore, it directly shows which regions were responsible for which classification as no spatial information was lost as no fully connected layers are used. This allowed for segmenting the image based on the classification an example of which is shown in Figure 14.

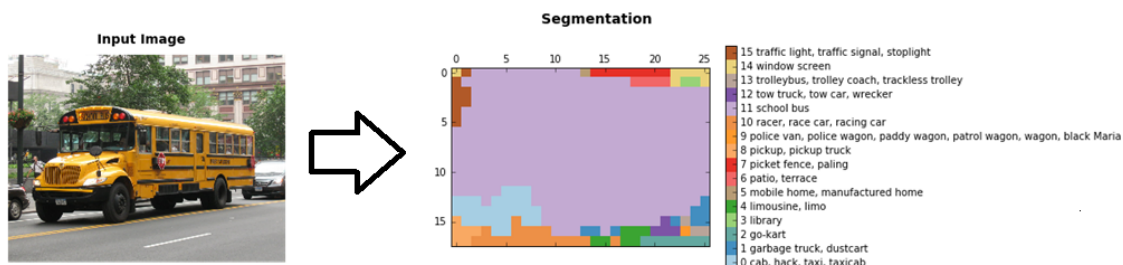


Figure 14: FCN output before upsampling, image from Pakhomov (58).

However, the resulting segmentation is a downscaled version of the image, which is a side effect of the convolutional layers. To get a more accurate image they upscale the image. However, they found that this resulted in very coarsely segmented images. To improve results they suggested using earlier convolution layers in creating the segmented images, as these earlier layers are less down-scaled. They then upsampled all these images and combined them using the lower-level layers as a way of adding detail to the segmentation images. This resulted in accurate semantic segmentation.

Following this Simonyan and Zisserman (16) suggested increasing the depth of the current FCN networks by reducing the size of the convolutional filters to a very small 3×3 . This allowed them to increase network depth of the FCN networks allowing for higher accuracy, as commonly was found that the deeper a network the better performance could be acquired.

As the performance of hardware increased the ability to generate deeper and deeper models became available. However, a problem was found when making very deep networks: When a deep network starts converging the accuracy gets saturated after which it degrades fast (15). This means that while increasing the number of layers increasing the expensiveness of the network, if the network does not need this expressiveness the last layers of the network have to learn the identity function. Learning the identity function is a problem in later layers as the network is not trained as a whole but one layer at a time. Furthermore, with increased depth the gradient of the network slowly vanishes with ever additional layer as the impact of a neuron on the outputs reduced with every layer i.e. the vanishing gradient problem.

To combat this Kaiming He et al. (15); introduced deep residual networks, where the network consists of so-called residual blocks where the input of this block is added to the output of the block. Now, if the network needs to learn the identity function, it is enough for the residual blocks to output zero which Kaiming He et al. expected to be easier than learning the identity function. These residual blocks are the key building blocks of the Resnet network Kaiming He et al. introduced. With these residual blocks, they were able to build a network much larger than was possible before encountering the vanishing gradient problem as the magnitude for the neurons further back in the network is increased by these blocks, which increased the overall performance. Using these Residual block they suggested multiple differently sized networks e.g. Resnet 18, Resnet 34, Resnet 50, Resnet 101, Resnet 151 where the number indicates the number of convolutional and dense layers in the network.

This was followed by R-FCN introduced by Jifeng Dai et al. (11) which proposed a Region proposal network(RPN) build on top of resnet 101 to enable bounding box generation. As resnet is an FCN and thereby has no FC layers it is considerably faster than Faster RCNN while having less overall complexity. To cooperate this RPN a Region layer is added after the last residual block of resnet 101 network which performance region proposals. These region proposals are then evaluating by using Positive-Sensitive Score Maps (PSSM) these maps give the pooled response for a certain region within the image. The generated region proposals are then overlapped with the PSSMs and if each region within the region proposal has a high pooling score, the region proposal is accepted.

Finally, to allow for instance segmentation Mask r-CNN was introduced as a flexible framework by He et al. (12). This framework added object instance segmentation to Faster R-CNN, which allowed objects to be more precisely labeled and masked on an individual basis. To achieve this, they combined two existing methods. To achieve semantic segmentation they used FCN; to detect objects they used Faster R-CNN. As Faster R-CNN can detect the individual object by bounding boxes and FCN can semantically segment the image, they suggested combining these two methods to allow for instance segmentation.

Even with methods specialized on instance and semantic segmentation, it is still considered to be expensive to create large enough training sets with correct labels for these cases. Instead of just focusing on making the algorithms faster, the focus shifted on using different types of training. Instead of relying on supervised training some focus shifted to unsupervised training and weakly supervised training.

A current popular way of reducing the cost of labeling a dataset is a method called weakly supervised learning. Weakly supervised learning in computer vision refers to the task of learning on a training set where labels are less precise than the desired output also known as weaker labels. This form of labeling allows for the easier creation of training sets as the manual annotation process is much less time-intensive e.g. creating specific masks for all objects in an image is much more resource and time expensive than just stating what objects are visible within the image. It also allows for the automatic creation of datasets as for instance search engines can now easily be used as a way of finding images with a certain tag (4; 23). This allows for the combining of multiple different existing datasets to automatically create a training set such as Zadrija et al. showed was possible with Street view footage and map information (5). In their paper, they labeled street view footage with map information and used the temporal aspects of videos to help with the classification. We use a similar approach, like Zadrije et al. using the information on maps to label a dataset. However we differ in approach and scope, where we use images instead of video and investigate if a more broad approach is possible for roadside object detection.

2.2 Unsupervised Learning

In unsupervised learning, no class labels are added to the training set. So in unsupervised learning, instead of trying to directly classify an object, the goal is to distinguish properties and find correlations and structure in the data (25). Unsupervised learning has seen some use for object detection problems as a way of bypassing the cost of creating large labeled datasets.

A recently popular technique used in unsupervised object detection is called zero-shot learning. In zero-shot learning, a model is pre-trained on normally labeled data and uses the features learned to differentiate new objects in unlabeled images based on these features. It looks at the learned features and groups images and objects based on the features present.

Methods like zero-shot learning exploit the recently created, large public datasets like Imagenet (44) and Microsoft's CoCo (43), while not requiring any labels for new objects (27). While this method already showed promise, a further improvement was made by Lampert et al. (28) who suggest using a model pre-trained on attributes, such as color, size, but also distinct objects like horn or tail, instead of image labels. With these attributes a better distinction could be made on unseen objects, leading to an increase in performance.

Another unsupervised solution for object detection was introduced by Wang et al. (26). In their paper, they introduce the idea of visual tracking in videos as a form of supervision which could be derived from the videos themselves. Instead of requiring hundreds of thousands of labeled videos for object detection, they only use unlabeled videos and the assumption that two images connected by time have a similar visual representation as they probably contain the same object only in a different time-frame. By using domain knowledge, they got close results to those of supervised solutions and were able to cluster similar objects together. Showing that object detection for new objects can be done in certain cases without creating new datasets, instead exploiting domain knowledge present within videos.

2.3 Weakly Supervised Learning For Object Detection

Weakly supervised learning (WSL) in object detection refers to the task of learning on a training set where labels are less precise than the desired output also known as weaker labels (see Figure 3). Using weaker labels has the key advantage that they require less time to annotate and in some cases can even be automatically generated by reusing and combining other existing datasets. WSL tries to use this advantage of easier to create labels/datasets without losing the ability for higher level annotations i.e. bounding box generation, semantic segmentation and instance segmentation. This could drastically reduce the cost of learning object detection and segmentation as the creation of manually labeled training sets can be exceedingly expensive. A downside of the method is that the information missing from the labels has to be acquired via different means i.e. domain knowledge, larger dataset, other priors.

The goal is thus to get extra information out of images and to rely less on the label. An example of an automatically created training set was shown by Crowley and Zisserman (4). They proposed to use Google images as a repository of positive samples for classification of new object classes to be able to sort a database of paintings by what objects they contain e.g finding all paintings that have dogs in them.

In their system, a pool of negative samples was prepared. Then, when a new search object query is sent, such as “dog”, their system sends this query to Google images to retrieve 100 possible samples, and together with preprocessed negative samples, a classifier is trained on-the-fly and applied to the database of paintings. As the Google images had no information about where in the image the dogs might be, the only label given to these images were image-wide. Crowley and Zisserman showed that the creation of a weakly supervised dataset can be easily automated for small tasks and can be done on the fly when needed.

While this shows promising results, one problem weakly supervised learning introduces when starting with image-wide labels for object detection and segmentation is localization. As image-wide labels have no location information available, there is in uncertainty about what and where the label refers to. This localization problem is usually solved via Multiple instance learning (MIL) (5; 10; 20; 21; 24; 30; 32) or localization via CNN (13; 19; 22; 34–36), where MIL is prominently used for object detection (bounding boxes) and localization via CNN for semantic segmentation.

A third, less-used approach also exists, which involves using saliency. Saliency is the measure of how much an object stands out in regards to its surroundings (75). Saliency-based methods show regions of interest or at least regions that stand out, which is usually indicative of an object being situated there. Although less used it can be incorporated into MIL methods (48) and localization via CNN methods (52).

2.4 Multiple Instance Learning

Multiple instance learning (MIL) is a method of labeling data by groups called bags. Instead of each object receiving a label the entire group/bag is labeled. If a bag contains only negative objects for a given label it is labeled negative. If even one of the objects in the bag does contain the label then the entire bag is labeled positive for containing that label. Figure 15 illustrated the difference between MIL and normal supervised methods.

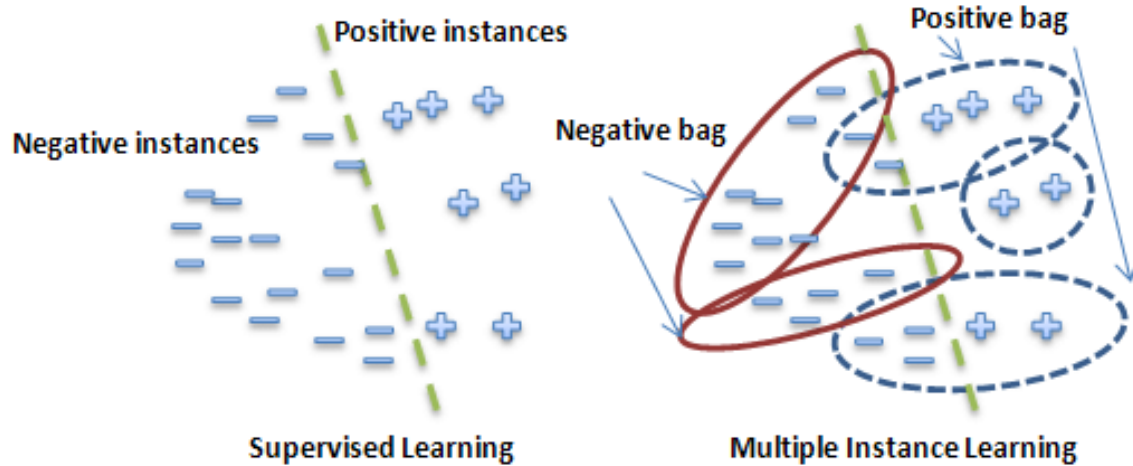


Figure 15: MIL based approach versus a supervision based approach, image from Kumar et al. (77)

MIL for weakly supervised object detection is usually used slightly differently; instead of a bag consisting of multiple images, each bag consists of multiple regions of a single image (21). This converts the problem from a localization problem to a MIL problem. The standard MIL pipeline for object detection usually consists of three major steps: Setting up the bags, initialization and finally localization and training a classifier (54).

2.4.1 Setting Up The Bags

The first step in the MIL process is to split the image into regions. Multiple methods exist for achieving this, as most region proposal algorithms can fulfill this step. Methods like the sliding window approach and selective search (7) have been used (55).

More advanced region proposal techniques like using saliency have also seen usage in splitting an image into regions (48).

A saliency-based visual attention model was proposed Itti et al. (47) with the goal of rapid scene analysis. In their paper they try to find regions of interest by creating a saliency map, their approach was inspired by how human see. They noted that the feature integration theory very adequately explains human vision. This theory states that human vision is first decomposed into a set of topographical features, after which these features compete for saliency. For their model, they followed this approach: First, they used three distinct types of features, color intensity (how high are the color values), color difference (what difference is there between groups of pixels) and local orientation information (how does orientation change pixels values respectively). They then use a spatial invariant approach that splits each feature into nine differently sized maps.

They then normalized these maps and apply a technique called center-surround in an effort to find outliers. These maps were then combined into a single saliency map were every feature has its own saliency per channel (see Figure 16). In this combining of maps, the features in a sense compete

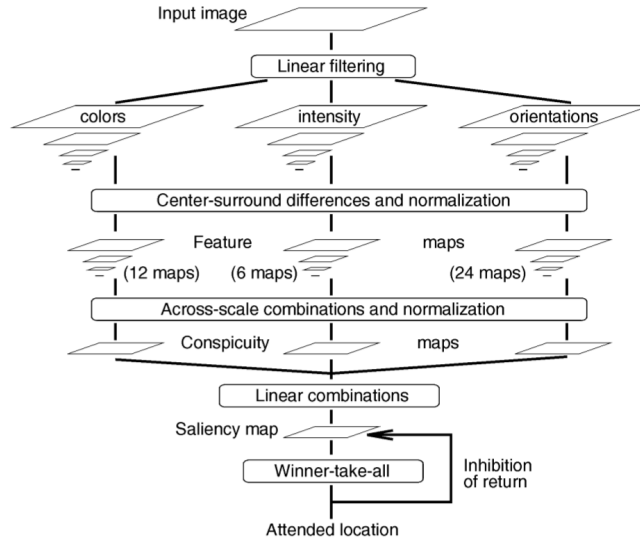


Figure 16: Saliency based visual attention model, from Itti et al. (47)

for saliency. Allowing for the prioritization of relevant information. Their result showed the ability to find disruptions in images even when noise was high.

2.4.2 Initialization

While some methods go straight to training after setting up the bags (49; 50), most methods do some additional preprocessing. Most methods reduce the size of the original images by some margin, with the intuition that the object labeled will probably be near the center and not cut off by the image boundaries, however this only works for pictures taken by people and not for autonomous cameras (31; 55).

A more advanced initialization was proposed by Song et al. (32). They noted that a large performance increase could be obtained by adequate initialization. After separating each image into a bag of 2000 region proposals they used a pre-trained R-CNN network to create a feature space of each region proposal. Using a nearest neighbors algorithm, they generate a list for every region proposal which contains the distance to all other region proposals, in terms of feature space. After which they selected all the positive bag region proposal that are relatively close to other positive bag regions, but far away from regions present in negative bags (see Figure 17). The intuition behind this is if a region is closely related to one of the negative bag examples then it probably does not contain the object as it is also present in negative examples. This method allowed them to already do a pre-selection of regions to learn from. The final step of the pipeline is to train an support vector machine (SVM) or in more recent method to train a CNN.

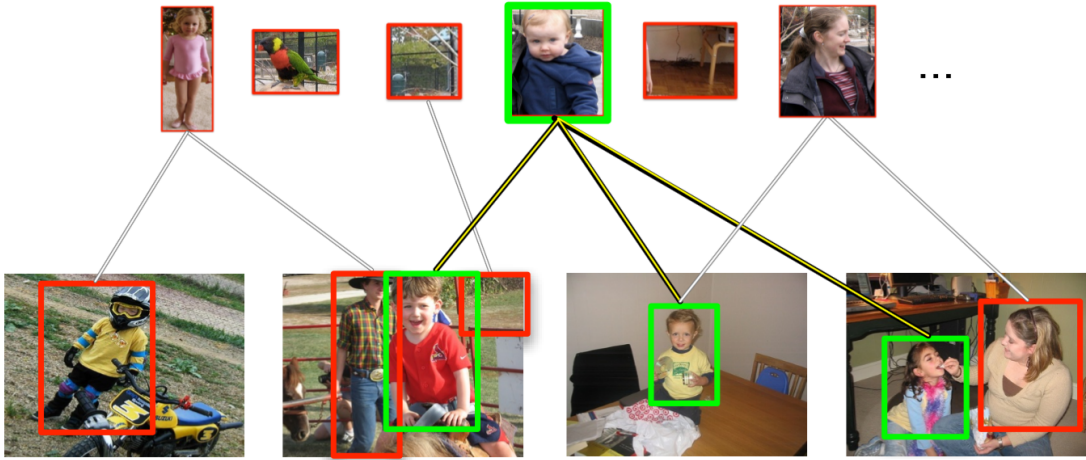


Figure 17: The green class is only connected to other green (positive) classes, from Song et al.(32).

2.4.3 Localization And Training Via SVMs

Support Vector machines (SVMs) are supervised learning models that try to differentiate and classify objects between two classes. They do this by looking at the different examples of both classes and trying to find a line that separates these two classes best (46) i.e. find a vector x that maximises the distance between two classes (see Figure 18).

Andrews et al. (39) introduced two SVM approaches for MIL, called MI-SVM and mi-SVM. MI-SVM and mi-SVM are different from SVM in that instead of differentiating all negative points from all positive points, mi-SVM and MI-SVM try to differentiate all negative points from at least one point from each positive bag i.e. mi-SVM and MI-SVM want to differentiate objects that are present in negative bags from objects that are only present in positive bags. mi-SVM simply tries to find the vector that maximises the margin between all negative bags and at least one point from each positive bag over all the possible labels assignments. MI-SVM instead tries to find the maximized margin within a bag i.e. find two sets of points where the margin is maximized for each positive bag. These two sets of points within a bag are then used to limit the search space when picking possible label assignments when finding the vector that maximizes the margin between the two classes. This method tried to exploit the fact that every positive bag should have at least one region that corresponds to the label. However, this approach seems to have problems with noisy data, as when a single positive example is present in any of the negative bags all the positives are now present in a negative example resulting in a big loss in performance. To solve these issues, more noise-robust method were suggested such as using CNNs.

2.4.4 MIL Localization And Training Via CNNs

Songs et al.(32) method showed that CNNs could be used to solve MIL problems (see 2.4.2). Wu et al. (40), proposed a weakly supervised MIL learning framework using a deep convolutional neural

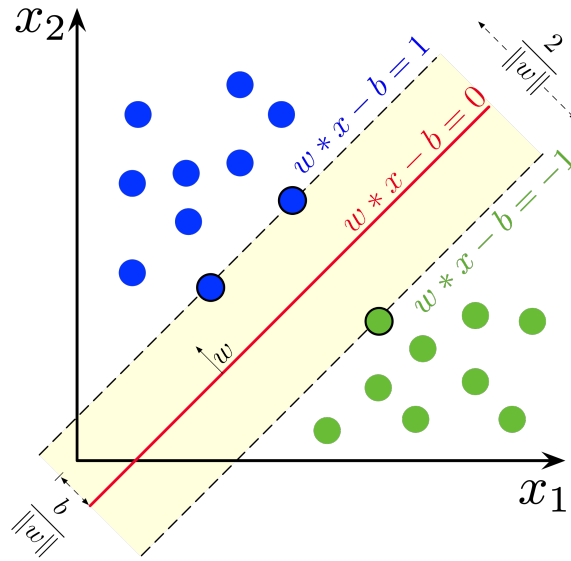


Figure 18: SVM differentiating two different classes trying to maximize w , image from (71).

network called deep-MIL. They used an Alexnet CNN (1) where they redesigned the last hidden layer to facilitate MIL. This layer combined the results of all objects in a bag i.e. they let every region in the bag individually pass through the network starting from the first convolutions layer to the last fully connected layer. All the outputs from the last fully connected layer are then combined with a softmax layer. The intuition behind this is if every region proposal within a bag is passed through the network and then combined at the last layer using a softmax layer, it should find the region that contributes the most for a given classification.

The loss of each bag is then aggregated over that bag and the lowest loss in each bag is used to calculate the gradient. The intuition behind this step is to find the region that has the lowest loss and use that to classify the entire bag. This works as not all regions in the bag should contain the object in them and we should only take the region that contains the object (see Figure 19).

While certain bags at the start of learning might classify the wrong region as correct since the gradients of all bags are combined as long as most bags classify the correct regions it would still eventually lead to the ability to learn. A more direct solution to solving the problem of wrongly classifying the wrong regions at the start of training was proposed by Sangineto et al. (51). Their idea is to first start with a simpler subset of images that are more reliable and only, later on, introduce the harder images. This would deal with the problem of wrongly classifying samples at the start of training. While dividing the dataset based on difficulty is not a new idea, it is new for deep network-based classifiers in end-to-end training pipelines (51). To create this simpler subset, they propose a self-paced iterative approach, where every iteration is using the current training state of the classification network to find the images with the highest classification. These images are then used for training the network. However, this process would not work for the first training cycle as, at this point, the network has no knowledge of the labels. To solve this, they propose a two-step procedure for the first iteration of the self-paced iterative approach.

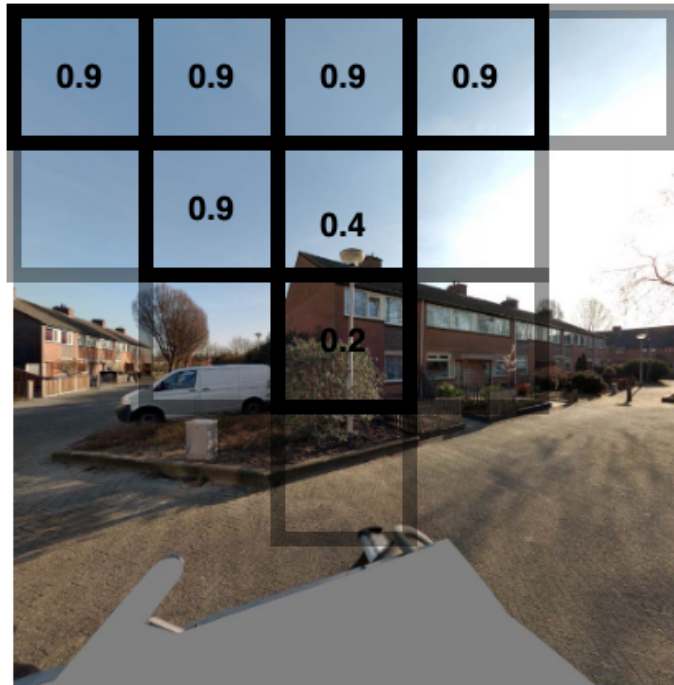


Figure 19: Example of how each region has its own individual loss and how the loss is lower for the location where the objects is.

First, they take a pre-trained CNN and change its softmax layer to be compatible with the training set i.e. set the amount the output classes equal to the number of labels present within the training set. Then using a small subset of the training set, they train the CNN using image-level labels. Using the weights of this network, they initialize a Fast R-CNN network, this Fast R-CNN need bounding boxes to be trained so they use the CNN to generate pseudo-ground truth bounding boxes for training.

This Fast R-CNN is then trained for one iteration as a sort of one-shot MIL solution. The result is the starting point of the self-paced iterative approach.

2.5 Localization Via CNNs

One method of solving the localization problem exploits the way CNN classify objects. After the CNN classifies an object, the key idea is to look back through the CNN layers to see which areas led to a positive classification. But as Fully Connected (FC) layers remove the ability to localize objects, Zhou et al. introduced using a global average pooling (GAP) layer to replace these FC layers (34). This allowed them to create Class Activation Maps (CAMs) that show the location of the features that led to classification. This GAP layer performs average pooling on every convolutional feature map resulting in a 1 by 1 layer for every convolutional feature map. This approach is similar to the approach of Long and Shelhamer et al. (53) but instead of using multiple sets of convolutional layers to construct a semantic segmentation map, Zhou et al. used only this GAP layer, which in a sense does something similar i.e. replace the FC layers and take features from multiple convolutional layers. Where their methods differentiate is that Zhou focuses more on the weakly supervised aspects while Long and Shelhamer use supervised training.

This GAP layer allows for the most discriminatory areas of the object to be found, as no location information is lost. However, less discriminatory areas are lost, as they are averaged by the GAP layer. This introduces a problem as only the most discriminative features remain leading to scarce CAMs where not the whole object is segmented, but only the most discriminative parts, e.g. Figure 20.

To solve this Grabcut can be used to create a more complete mask/segmentation of the object. Rother et al. (74) suggest the method Grabcut as a way to go from a bounding box or incomplete mask to a more accurate mask.

Grabcut grows the region we consider to be correct or incorrect i.e. foreground and background, where foreground is a region where the object is present and background a region where the object is certainly not located. This growing is achieved using a graphcut algorithm which uses a Gaussian Mixed Model (GMM) to find the optimal location to segment the images. A GMM is a probabilistic model that in this application models the probability that two regions are located next to each other. If no foreground information is available, as in the case of only using bounding boxes, this background region growing is the only thing used to refine the bounding box.

Ahn and Kwak proposed to solve the discriminative problem differently by adding a DNN (deep neural network) called AffinityNet (35). This AffinityNet is then added as a post-processing step, and tries to predict features next to the boundary of the CAM. If it does this successfully, that area is added to the CAM. The intuition behind this is simple: If the AffinityNet can correctly

Figure 20: CAM only detects the face of the horse, not the rest of the horse. Figure from Ahn et al. (35).



predict a feature then this features was apparently near the CAM maps often enough to allow for this prediction to happen i.e. features that are often present should probably be included.

Going in more detail, AffinityNet is trained to predict class-agnostic semantic affinity between two adjacent coordinates. This affinity is predicted on feature maps, as to increase computational efficiency, where the semantic affinity between two feature maps is the difference between each feature in the feature map. The AffinityNet is trained by using areas within CAM as training examples, where areas with very high confidence are used as positive examples and areas with very high confidence of being background are used as negative examples. They specifically avoid areas with medium confidence, in both positive and negative examples, as these are too risky to use as training examples. These training examples were then used to train the AffinityNet. After training, the predicted affinities are used in a random walk algorithm as probabilities. Regions with higher affinity have a higher chance of being added to the CAM.

Another approach to solving the problem of a scarce CAM was introduced by Wei et al. (52). They proposed to use an adversarial erasing approach in which they remove the most discriminative parts of the CAM after classification. With the idea that now without the most discriminative part in the image if we retrain the network on these images we would find the next most discriminative parts of an object. They repeat this process until classification scores drop below a certain threshold. To make sure no classification happened on the removed part of the image, the pixels were replaced by the mean value of all training images.

The resulting CAMs from this method are then combined as weighted maps, taking their classification scores as weights. To increase performance, they also sought to use background localization cues, as this was found to increase performance when semantically segmenting an object.

To accomplish finding the background, they used a saliency-based method. This returns areas of interest, thereby also showing areas of disinterest. Then, when generating a mask, they ignored this background as well as three other kinds of pixels: pixels that were in removed foreground regions, pixels that lie within object CAM maps, and finally pixels that are not assigned to semantic labels.

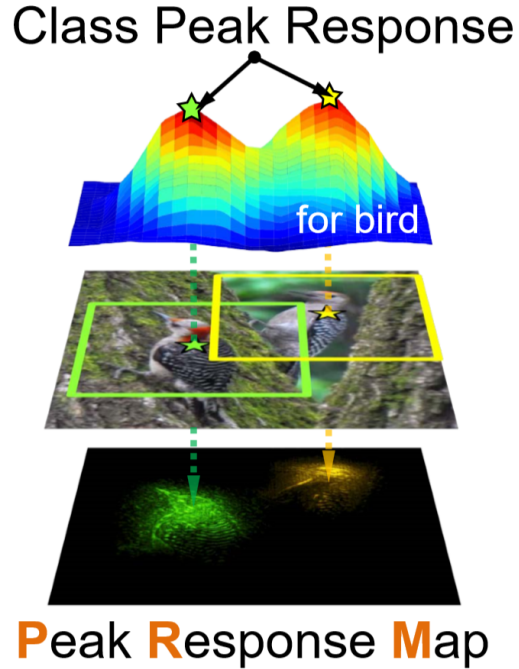


Figure 21: Peak response maps from Zhou et al.(13).

While these methods allow for semantic segmentation, instance segmentation is not possible, as the network can only tell which features lead to classification, but not to specific classes. However, Zhou et al. (13) found that peaks in the class response maps usually correspond to instance-related visual cues inside individual objects. When multiple of these peaks exist in an image, this can be indicative of multiple objects being in the image as shown in Figure 21.

Zhou et al. (13) proposed to use a pre-trained FCN where a peak stimulator layer is added before the softmax layer. This peak stimulator tries to stimulate peaks that are closely related to other peaks within the class response map. Once training is completed, peaks that have been found are transformed into peak response maps by looking which regions led to the creation of this peak. This Peak Response Map (PRM) is then combined with the CAM and with an off-the-shelf segmented proposals to provide instance-level segmentation. The performance of this approach is however bounded by the quality of the off-the-shelf segmentation proposal.



Figure 22: Example of BGT map.

3 Resources And Materials

This section goes over the available resources and materials in detail. First, we discuss the maps used providing semantic information (Section 3.1). After which, we discuss the street view images used in Section 3.2. Finally, we discuss over the depth images used in Section 3.3.

3.1 Maps

In this thesis, will focus on two maps the BGT (basis grootschalige topografie)⁴ illustrated in Figure 22 and the DTB (Digitaal Topografisch Bestand)⁵ illustrated in Figure 23, which are both dutch maps that show a large number of objects and region within the Netherlands (56).

Where the BGT map contains all municipality-owned objects e.g. all objects near and within cities, villages and on smaller roads, and the DTB map contains all state-owned objects e.g. all objects near and within highways, large provincial roads and waterways. As the bounds between these regions can sometimes be vague, the maps overlap on occasions as seen in Figure 24.

The objects present within these maps are divided over three main types of objects: line objects, region objects and symbol objects. Where line objects are objects like road markings, hedges, guardrail, etc, and where region objects are objects like cities, fields, roads, buildings. Finally, symbol objects directly refer to single objects by giving a single location. Example of these are objects such as street lights, trees and traffic signs. These symbol objects are the objects we are most interested in and contain most street relevant objects. The objects within the symbol type are subdivided into 32 classes, where each class has a certain theme of grouping (for the full list see Appendix 9.4). For example, the class pole contains all the pole type objects, see Table 1.

⁴ Dutch for basic large scale topographical map

⁵ Dutch for Digital topographic file

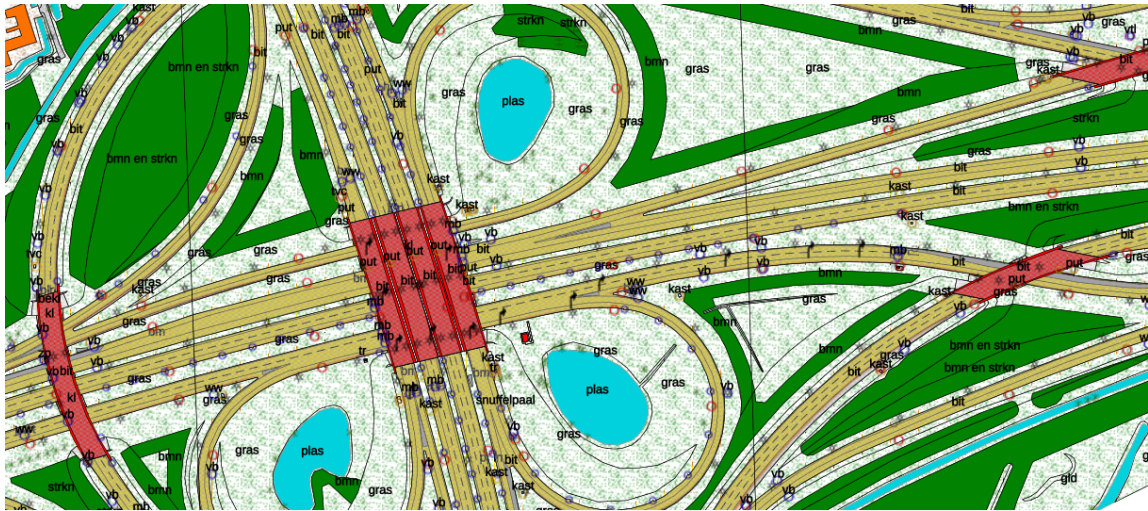


Figure 23: Example of DTB map.

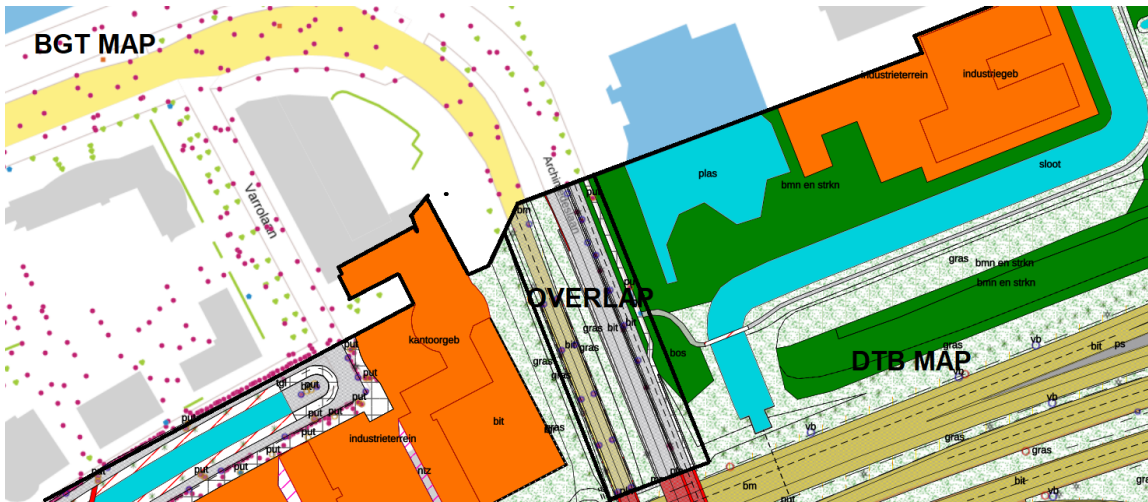


Figure 24: Difference between BGT and DBT maps and a region where the overlap.

Object name	Number of entries
Sirene (Sirene)	189
Afsluitpaal (barrier post)	510277
Verkeersbordpaal (traffic sign pole)	197381
Portaal (traffic height restrictive pole)	3113
Drukknoppaal (button pole)	5558
Poller (raising barrier post)	4007
Telpaal (counting pole)	286
Hectometerpaal (hectometre post)	35952
Verkeersregelinstallatiepaal (traffic control installation pole)	39653
Grensmarkering (border marking)	2768
Lichtmast (street light)	1987249
Haltepaal (busstop pole)	6727
Praatpaal (emergency telephone)	285
Vlaggenmast (flag pole)	13677
Dijkpaal (dike pole)	483

Table 1: Number of objects within the class "Paal" (Pole) for the entire Netherlands

While certain objects only occur a few times in the entire dataset, for example Sirene is only present 189 times, Objects like, for example, street light are present almost two million times. These maps not only give location information about objects, they also give the time of registration which specifies when an object was mapped. Furthermore, the location information is highly accurate usually within one meter. This makes using these maps ideal for our problems as they have large amounts of different objects and most of these object also have a large number of individual entities illustrated by Table 2. See Appendix 9.4 for a full list of available objects.

3.2 Street View Images

To find images for the location that the BGT and DTB maps provide, a large set of available street view images of the Netherlands is required. To acquire these, we will use Cyclomedia's repository of street view images as they have mapped almost all roads in the Netherlands with high-quality images, e.g. see Appendix 9.3. These images are created by a camera system with five cameras that all take one image every five meters. These five cameras are pointed in different directions (up, forward, left, backwards, right) allowing them to be combined to form a single cyclorama. Where a cyclorama is a 360°images bubble that is viewed from the inside, Figure 25 illustrates a 2D representation of a cyclorama. The resolution of these image is normally 1536×1536 but this is dependent on when an image was taken e.g. images taken before the year 2016 and by an older system are 512×512 . However, for our research, we will only rely on the latest images which all have at least a 1536×1536 pixel resolution.

The camera system is normally situated on a car as illustrated in Figure 26 this allows it to be driven around roads, bicycle paths and squares. The system also contains a GPS sensor to measure the location of each cyclorama to be recorded. This allows each image to be position on a map and requested by a location. Images can also be request given a specific image id or by a region

BGT-object	Amount	DTB-object	Amount
Drain	2550088	Map height marking	1124343
Street light	1983625	Illumination object	255984
Manhole	1266851	Drain	236510
Barrier post	509430	Traffic sign without cable	144624
Traffic sign	229045	Bollard pole	106911
Traffic sign pole	197055	hectometre sign	82356
Fire hydrant / well	170810	Manhole-cover	79874
Playground equipment	105821	Bollard rock	55370
Bicycle rack	105265	Utility cabinet	54059
Garbage can	94533	Roadside protection block	31168
Water well	87745	Paint sign	27876
Bench	83727	Signpost without cable	25469
Waste separation can	52847	Traffic light	20160

Table 2: Top 13 most occurring objects from the BGT and DTB map. For a full list, see Appendix 9.4.



Figure 25: 2d representation of a cyclorama.

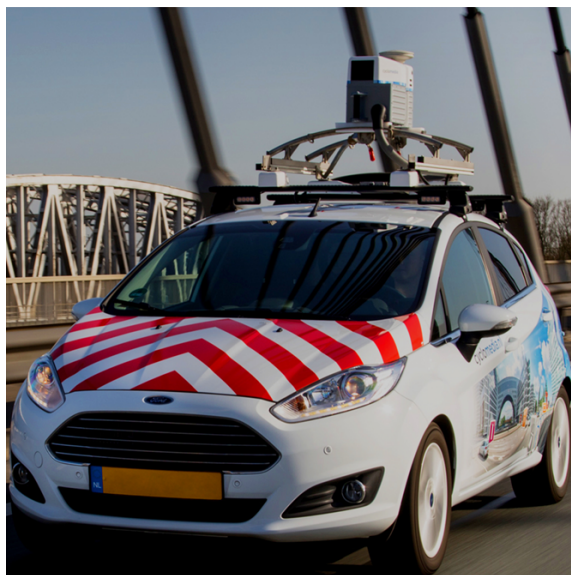


Figure 26: Cylcomedia's camera system.

in the form of a polygon in which all images within are returned. It also allows for the sorting on a specific year e.g. we might only want recently shot images since an object was only recently mapped. Next, to returning these five images, some extra information is also passed namely: an image id, location information, the direction of the car relative to north and finally a precision measure that shows how accurate the location and direction parameters are and how much they can be off, due to uncertainty in GPS information.

3.3 Depth Images

For some regions depth information is also available, see Appendix 9.3 for the availability of depth information. This depth information is created by a lidar system ⁶ which is also placed on the camera system. This lidar systems generates a set of point together with the five camera every five meters. However, this lidar system has some lag compared to the camera system which in some case can introduce a minor shift where the generated point do not exactly match the image. This is however minor and small shift and should not have any large scale impacts.

Once a set of points is generated by the lidar system for a specific location, these point are transformed into a mesh were point close to each other are connected. This mesh can than be transformed into an depth images where the distance is encoded within the color of a specific pixel in an image as illustrated in see Figure 27. For regions where no points where found the distance is set to -1, to indicate no depth information is available for that region. This could be due to there not being

⁶ A lidar system is a laser system that sends out signals of light and measures the time for the signal to return and as the speed of the signal is known, this can be used to calculate the distance to a particular point.

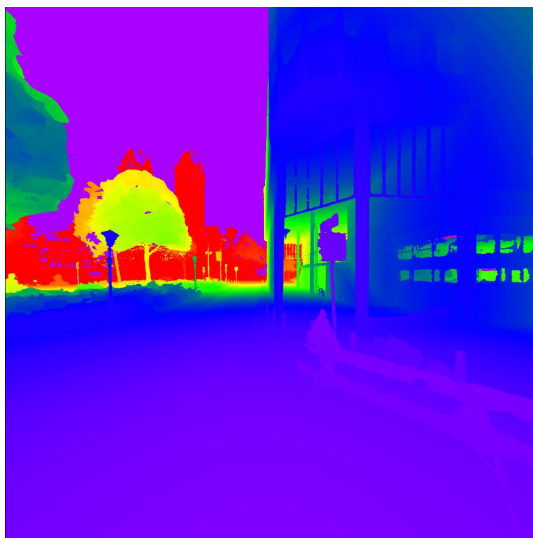


Figure 27: Example of depth image, where the color of a pixel represents the distance from the camera to that pixel.

any object e.g. the sky or due to a particular surface refracting the incoming laser signals from the lidar system. These depth images can be requested the same way street view images are requested.

4 Methodology

In this section, we will go over our methodology for answering our research questions. First we will discuss our method of automatically generating a dataset from maps and images (see Section 4.1). Second, we will go over our method for training on this automatically generated dataset for both image-wide object detection, bounding box object detection and semantic segmentation (see Section 4.2). Finally, we will go over our method of improving the results by using automatically generated masks (See section 4.3).

4.1 Automatic Dataset Generation Method

An overview of the automatic training set generation method can be seen in Figure 28. To automatically generate a training set and validation set first a selection of classes needs to be made. This selection of classes can be any of the objects within either the DTB or BGT maps (see Section 3.1). However, for our method, we only investigated using points objects thereby disregarding line objects and region objects. We make this decision as these other types of objects have different kinds of problems associated with them i.e. how much of a line object or region object should be in view for us to consider it visible. Furthermore, most of the line and region objects are used to define boundaries, districts, and paths while our main focus is on roadside objects.

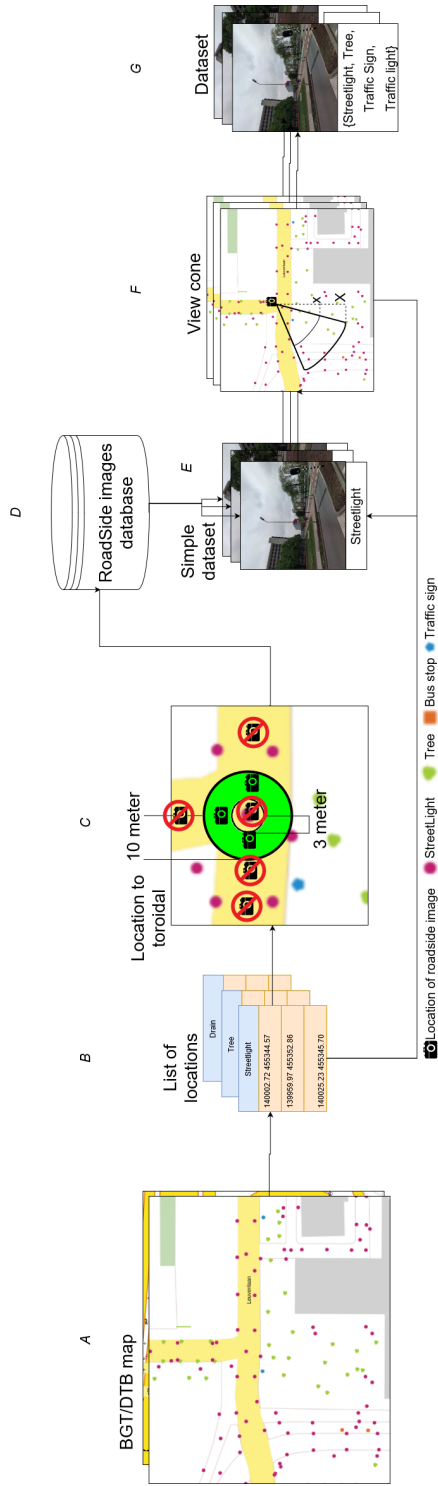


Figure 28: Overview of the automatic dataset generation method. First, we take the BGT and DTB maps(A). These are then transformed into a list of coordinates of objects (B). A toroidal region is described (C) on the map around the coordinates of objects with a range of three to ten meters. For every object, all images in these zones are downloaded until the requested size has been reached. Figure 30 illustrates why objects closer than three meters may not be clearly visible resulting in them being ignored. The images selected are then downloaded(D), focused on the object and cropped to images with an FOV of 90°(E). The images are then labeled with the labels of the object that was used to the image. To find the other objects presented within the images a view cone (F) is constructed by using the location of the image, FOV, two pre-selected distances X_{min} and X_{max} , and the direction of the object. All objects within this view cone are then considered as being visible. Resulting (G) in our automatically generate dataset.

When making a selection of classes, similar classes between the two maps can be combined into a single class. This can be beneficial as in a few cases the class names between the maps might differ while they represent the same objects. For instance, the class ‘street light’ in the BGT maps is called ‘illumination object’ in the DTB map. While small differences are present within these two class labels, i.e. ‘illumination object’ also includes lights that don’t have a pole associated with them while the ‘street light’ class only contains lights that are on top of street poles, most locations within both maps point to the same type of object a simple street light. Furthermore, some objects are visually too similar for a clear distinction to be made e.g. Figure 29 shows the difference between the different types of Utility cabinets which can sometimes be very hard or impossible to distinguish from an outside perspective. Combining them in most cases is a more optimal strategy and when necessary to distinguish them one could use a second classifier specifically trained on differentiating these similar classes.

Finally, the size of the dataset needs to be selected which determines the number of images per class. The number of labels cannot directly be determined beforehand as it is unclear how many objects are visible within the background before processing. This size should be generally a lot lower than the objects with the least amount of occurrences in the DTB and BGT map, as there is no guarantee how many of those objects are close to streets/roads and how many images can be found per location.

Once a selection of size and classes have been made the automatic dataset generation can begin which is illustrated in Figure 28. First, we take the BGT and DTB maps(A) which we then are transformed into a list of coordinates of objects(B). A toroidal region is described (C) on the map around these coordinates with a range of three to ten meters. For every object, all images in these zones are downloaded until the requested size has been reached. A minimum range of three meters is used as objects closer than three meters may not be clearly visible as they have a chance of being cut off as illustrated in Figure 30. The images selected are then downloaded, focused on the object and cropped to images with an FOV of 90°(E). We selected 90°as this allows most object to be completely visible in most cases especially when including a minimum range of three meters. It also allows the cycloramas to be easily split into four sections reducing the overall complexity when making predictions.

The images are then labeled with the label of the object that was used to find an image. To find the other objects presented within the image a view cone (F) is constructed by using the location of the image, FOV, two pre-selected distances X_{min} and X_{max} , and the direction of the object. All objects within this view cone are then considered as being visible. However, finding a good, single cutoff point for distance could introduce large amounts of label noise. Therefore, we introduce a hard region to the view cone where we ignore the class for a particular image when calculating a loss if a class of objects is only present in the hard region of the view cone. This hard region does come at a cost as some of the images get falsely discounted from the negative set, decreasing the number of images that can be trained on. But as this negative set is expected to be several times larger than the positive set as most classes are not present within most images the expectation is that this trade-off will improve overall performance. The size of the normal label region of the view cone we denote by the distance X_{min} and the hard region by the distance X_{max} . Where a label is considered in the normal label region if within the view cone and the distance to the object is smaller or equal then X_{min} i.e. $x \leq X_{min}$ where x is distance. While for the hard region the distance is greater then X_{min} and smaller then X_{max} i.e. $x > X_{min} \wedge x \leq X_{max}$.



(a) Image of a CAI cabinet



(b) Image of a cabinet house



(c) Image of a cabinet house



(d) Image of a cabinet house

Figure 29: Images of two types of utility cabinets being very similar and very hard to distinguish from each other.

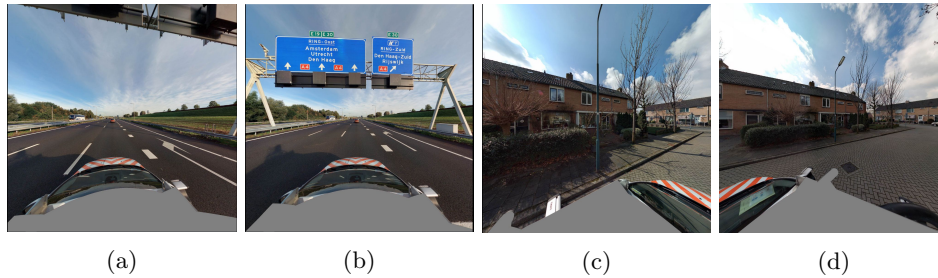


Figure 30: Difference of images selected when minimum range is present (b,d) and when not present (a,c). Illustrating that a minimum range can reduce objects being cut off by image boundaries.














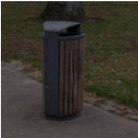






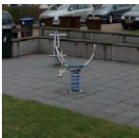

We investigate the performance of the method by selecting 20 objects chosen to be differently shaped, sized, rarity and environment they are usually situated in. Figure 31 illustrate our selected classes. With these objects, a training set of 16,000 images (800 images per class) and validation set of 4,000 images (200 images per class) is created. It should be noted that some of the chosen object classes actually consist as multiple different classes in the maps. These were combined as together as they were too similar to visually distinguish e.g. in the case of utility cabinet they consist of the classes ‘Cabinet house’, ‘Communication Cabinet’ and ‘Electronic Cabinet’⁷.

The decision to use 20 different classes comes out of a feasibility study we did (see Appendix 9.1), where we concluded that increasing the number of classes can increase performance as the number of negative images increases. Furthermore, increasing the number of classes can help with the localization of objects as the presence of one object can remove the uncertainty that another object is at that location. While this does not happen for all objects, it can improve the performance of objects that are never located next to each other e.g. a street light can certainly also have a traffic sign on it, but a street gully will never be near an electronic freeway sign.

To evaluate the automatically generated dataset, we manually annotate a set of 400 images from the dataset. We then compare this manually annotated set with the labels given by our method. Using this manually annotated set, we can also fine-tune the label-range with a given value X_{min} and X_{max} .

However, as this manually annotated set is normally not available, we suggest selecting values based on the size and visibility of the object, placing them into one of four ranges expanding the ranges when needed : Very Small ($X_{min} = 5, X_{max} = 10$), small ($X_{min} = 10, X_{max} = 15$), medium ($X_{min} = 20, X_{max} = 30$), large ($X_{min} = 35, X_{max} = 45$). Figure 32 show why size cannot be the only determination made when finding the range of the object as the general positioning of certain objects can greatly impact visibility. Doing this for our 20 selected objects results in Table 3.

⁷translated from Dutch ‘kasthuis’, ‘CAI kast’, ‘elektrakast’

Traffic light (verkeerslicht)			
Utility cabinet (kasthuis, CAI kast , Electrakast)			
Manhole(inspectie -/ rioolput, put deksel)			
Street gully (kolk)			
Bicycle stand (fietsenrek)			
Garbage can (afvalbak, prullenbak/vuilnisbak)			
Streetlight (lichtmast, verlichtingsobject)			
Camera (tv-camera)			
Roadside protection block (bermbeschermingsblok)			
Public play equipment (speelvoorziening)			

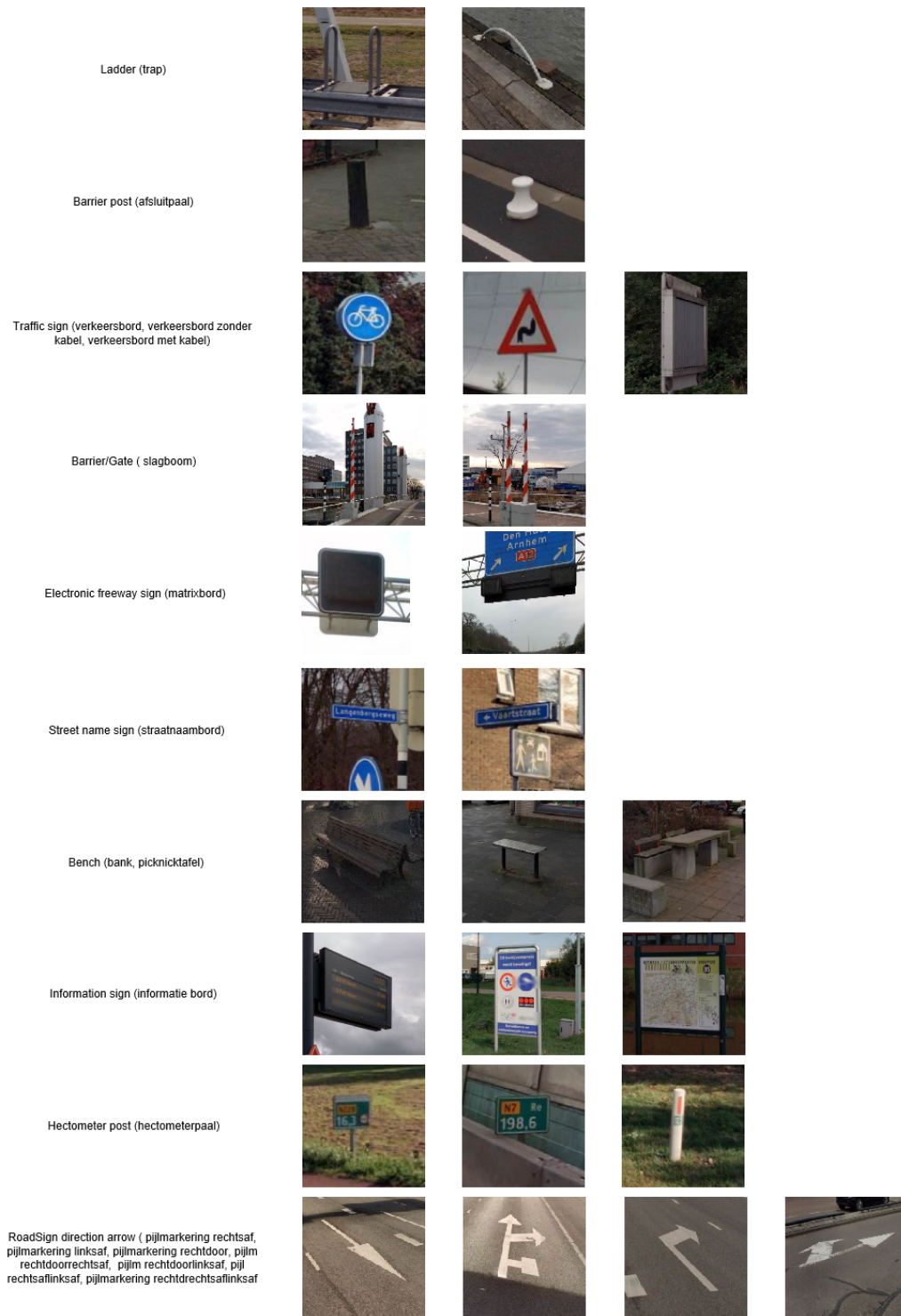


Figure 31: Examples of the selected dataset, names in brackets indicate original Dutch labels as given by map



Figure 32: Image showcasing that object in the ground are badly visible when viewed from a distance due to their size and the angle of viewing.

Very small (Xmin=5, Xmax=10)	Small (Xmin=10, Xmax=15)	Medium (Xmin=20, Xmax=30)	Large (Xmin=35, Xmax=45)
Street gully	Manhole	Barrier post	Street light
	Camera	Utility cabinet	Barrier/gate
	Street name sign	Bench	Electronic freeway sign
	Hectometer post	Garbage Can	
	Bicycle stand	Traffic light	
		Traffic sign	
		Roadside protection block	
		Information sign	
		Public play equipment	
		Ladder	
		Roadsign direction arrow	

Table 3: Table 3 The label-ranges of the different objects.

4.2 Training On The Automatically Generated Dataset

In this section, we describe how we train on the automatically generated dataset and acquire image-wide object classification results (see Section 4.2.1). We then describe our method of acquiring location information on the automatically generated dataset using CAMs 4.2.2. We follow this by describing how we acquire bounding box object detection results (see Section 4.2.3) and semantic segmentation results with the location information acquired by the CAMs (see Section 4.2.4).

4.2.1 Classification On The Automatically Generated Dataset

We investigate training on the generated dataset with a CNN network as these networks are generally more capable than non-deep learning methods especially when handling noisy datasets. For this, we will use Resnet 50 (15) as it is a run-of-the-mill network that is often used for object detection. Furthermore, the network is an FCN and as such can easily be converted into a CAM generating network, a key feature we intend to exploit to acquire location information.

We also investigated a VGGnet 16 network, as this was originally used for the CAM method. However, newer methods (13) show increased performance on the Resnet 50 network and we found similar results in overall classification see Appendix 9.2.

We use a pre-trained version of Resnet, which is pre-trained on Imagenet, as our base model. We then changed the last output layer of the network to a sigmoid dense layer instead of a softmax layer as our problem is a multi-label detection problem. We set the size of this last layer to be equal to the number of classes.

For the parameters of the network, we set a batch size of 4, due to memory constraints, and investigated multiple different learning rates. Furthermore, we used the decay and momentum values suggested by Khe et al. (15). We kept training until our validation loss stagnated for more than four epochs. Empirically, we found that after four epochs of no improvement, training longer will not increase the performance. If this was the case, the epoch with the lowest validation loss is selected.

As a loss, we use a binary cross-entropy loss, where we set the loss to zero for samples labeled ‘hard’ (-1), thereby ignoring the entire set of images for that class. We used binary cross-entropy, as it is a loss often used in multi-label classification.

We take the mean of the loss over the current batch resulting in a loss per batch. This is a normal step when using binary cross-entropy. Finally, we add class weighting to reduce the effect of class imbalance which is necessary as certain classes are alloted more prevalent in the street environment e.g. street light occur more frequently than cameras in the background of other images. The class weight simply reduces the network’s incentive to over-focus on the over-represented classes.

4.2.2 Localization Of Objects With CAMs

To be able to localize the object, we detect using our network we need a method that can acquire localization information from our weakly labeled dataset. Numerous methods for this exist but the two popular used ones are MIL based approaches and CAM based approaches. We chose to

use the CAM generation method as introduced by Zhou et al. (34), as this method has some key advantages over MIL based approaches.

First, MIL based approaches overall need very accurate labels and noise within these labels can significantly reduce performance as seen in Song et al. (32).

Second, MIL based methods require a method to generate bags, for this region/objects proposal methods such as sliding window and selective search. This makes the rest of the MIL based method heavily depended on the quality of these methods as they form the basis of the method. Furthermore, a sliding windows based approach would require a preset size to be determined for every object class. Within the road environment, this leads to problems as the distance from the camera to the object is highly in-consistent and so having a single fixed value would reduce performance significantly. This fixed-size also creates problems as inter-class variations within certain classes can be extensive e.g. while most utility cabinets have the same square shape they do differ in size as illustrated in Figure 29.

Third, most MIL based approaches are limited to bounding box generation as they lack the required precision for semantic segmentation. As one of our goals is to enable segmentation of objects, Zhou et al.'s CAM method fits our problem better.

Generating a CAM from a network like Resnet 50 is straightforward, as the network is already an FCN, thereby already maintaining localization information within the network. To now generate the CAM from this FCN, the network is first trained normally. After the network has been trained and an optimal epoch has been found, the network is changed to also output the values of the last convolutional layer. When selecting the last convolutional layer we have a choice: the last convolutional layer can be selected, which is located in the last identity block, or the last 'add' layer can be selected, which is located after the last convolutional layer and is a sum of the last two convolutional layers. Both layers can be used for CAM generation and we investigate performance of both methods.

Once a prediction is made, the output of the last convolutional layer or add layer has a shape of $(16, 16, 2048)$. For every class, the values that led to a positive classification need to be found. To accomplish this, the weights from the dense layer are taken, and their dimensionality is $(n_class, 2048)$. These weights can map the convolutional layers value to a certain label. By taking the dot product between these weights and the convolutional layer, the labels are directly connected to the convolutional map, resulting in a CAM for each class shaped $(n_class, 16, 16)$. Figure 33 illustrates this process and shows why the resolution of our CAM is directly linked to the size of the convolutional layer.

Once the CAMs have been generated, they are normalized and their values are fit within a range of 0 to 255. This makes it easy to visualize the CAMs. The normalization allows for a generic threshold to distinguish foreground and background, and to take the most activated regions out of an image. Furthermore, this normalization reduces the effect of confidence values on the CAM. We investigate multiple values for finding an optimal threshold to distinguish the foreground and background of the image (see Figure 34).

Looking at this process of CAM generation, it becomes clear that CAMs are in a sense the output of the network if one would remove the last merge layer. Therefore, one could also directly remove the merge layer, resulting in an output of $(n_class, 16, 16)$. To generate labels as well, one could use a merge layer outside the network, which would change the CAMs $(n_class, 16, 16)$ to labels (n_class) .

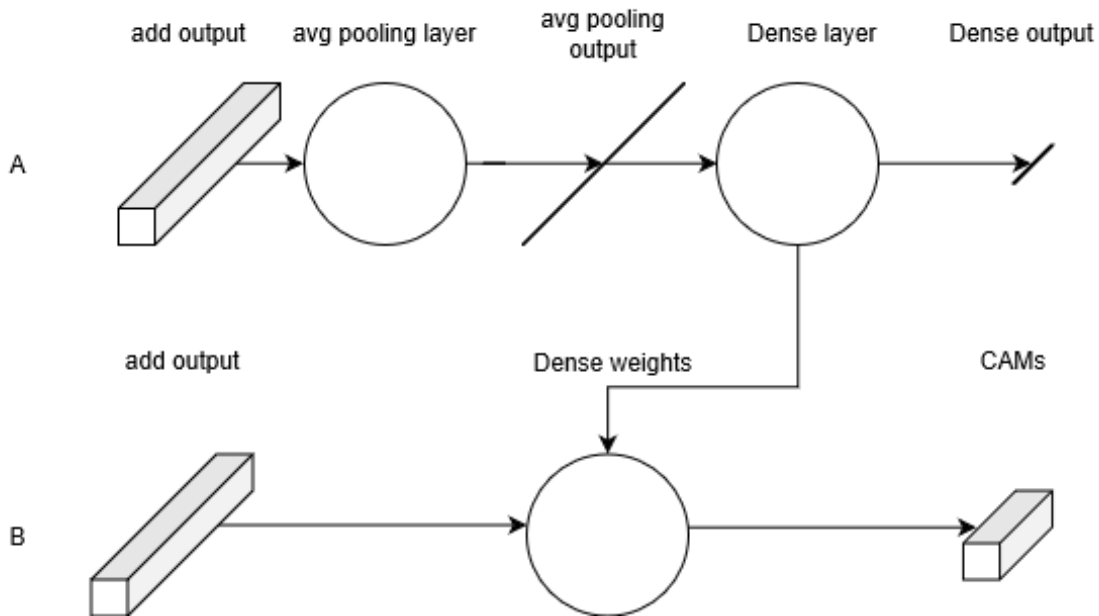


Figure 33: Image A shows a simplified diagram of the last layers of a Resnet 50 network, Image B shows how we use those layers to generated CAMs. Where the add output is the output of the last add layer, in image b this could also be replaced with the output of the last convolutional layer which has the same shape and size as the add layer. The weights that where used to transformed the average pooled layer output to the dense output are then reused to transform the add output or last convolutional layer output to CAMs for all the classes in the model.

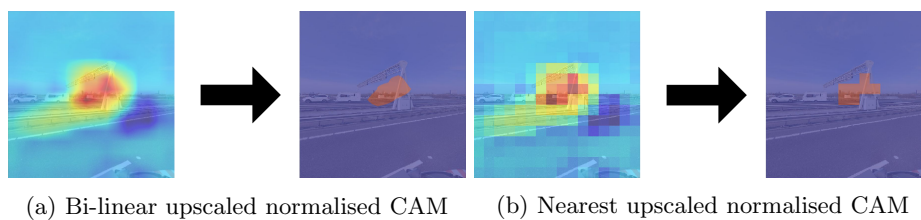


Figure 34: Effect of thresholding the normalised CAM on a threshold of 200

However, after preliminary research, we found that this network was incapable of converging to any real values for both the CAM and labels and therefore we did not investigate this path any further.

4.2.3 Bounding Box Generation

We investigate two methods that convert our CAMs into bounding boxes. Our first method simply creates a bounding box around the CAM at a certain threshold. Our second method using region proposals generated by a selective search algorithm (7) and scores these region proposals based on two features. The amount of foreground that is contained by the proposed bounding box minus the amount of background that is contained by the proposed bounding box i.e.

$$Score_x = \sum F[z] - \sum B[z] \quad (7)$$

where F is the foreground mask, B the background mask, z the proposed bounding box.

To generate the fore- and background two thresholds are used, the first selects the foreground, while the second selects the background. An intentional gap is left between these thresholds to allow for the growth of the bounding boxes in regions near the foreground.

To find an appropriate value to threshold the CAM we investigate multiple threshold values. To validate our outputted bounding boxes we compare them against a manually annotated validation set (see Section 5.1).

4.2.4 Semantic Segmentation Method

As with bounding boxes we use CAMs to help localize the object. We investigate two methods to go from the CAMs to segmentation masks. The first is to simply use a high threshold on the CAM and use that as our segmentation mask. The second option we research is to use GrabCut where we select the highest valued parts of the CAM as our foreground and allow it to grow to our background which are the lower parts of our CAM. We selected GrabCut as it is a simple algorithm to allow for semantic segmentation. To find an appropriate value to threshold the CAM we investigate multiple threshold values. To validate our outputted masks we compare them against a manually annotated validation set (see Section 5.1).

4.3 Automatically Generated Masks

In this section, we discuss our method of using available depth and map information to help guide our learning process and increase the quality of our CAMs. First will discuss how we use these form of information to create approximate masks (see Section 4.3.1). Second, we will discuss how these can be used to train our network (see Section 4.3.2). After which we will discuss how these masks can also increase the resolution of the CAMs (see Section 4.3.3).

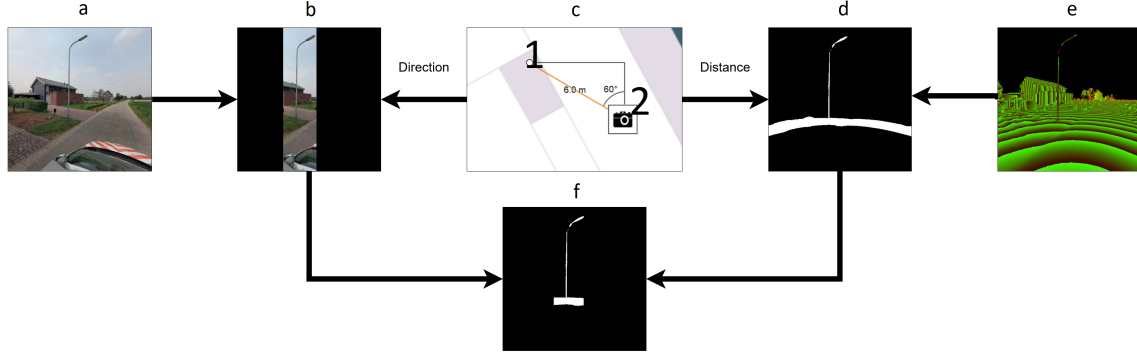


Figure 35: Overview of our proposed CAM loss method. First, to generate an approximate mask the direction of all objects of a certain class is first calculated. Then the distance to all these objects is then calculated. Using these distances a range is created with which all areas within the depth mask that fall within this range are found. This is then combined with a range over the direction resulting in approximate ground truth masks. This approximate mask ground truth mask is then compared using cross-entropy with the CAM which allows for a loss to be calculated.

4.3.1 Approximate Masks Generation

To improve the accuracy of CAM and help guide the learning process for bounding box detection and semantic segmentation. We investigate improving the CAMs by introducing an additional loss that aims to punish the CAMs during training when focused on the wrong part of an image i.e. CAM loss. With the intuition that even if a label is correct it should also be able to find the object within the image, otherwise it might be over-training or using the wrong feature for classification. As no direct object-masks are available within the dataset, an approximate ground truth mask is instead constructed out of the depth image (available from CycloMedia Lidar dataset), the relative direction of the object within the image (available from the object location and image location) and the distance to the object (also available from the object location and image location). By combining this information a rough mask can be constructed that while not perfect should be enough to point the CAM in the right direction(see Figure 35).

To create these masks, first an approximate depth range dr is calculated. Using the distance di between the object and the recorder's world location (image location), the range rw , in which a pixel is considered to be part of the image is selected. If the object is further away the size of the range is increased, since the distance measurement becomes noisier as the distance increases. Where the range has an increased size the further the object is away as the depth images measurements become more noisy as distance increases. Resulting in

$$rw_i = \max(di_i * o, 0.5), \quad (8)$$

where o is a variable greater than 0. To further specify the masks, a direction range dr for each object is also taken into account. First the relative direction of the object θ_d is calculated with regards to the recorder's world location b . Resulting in

$$\theta_d = \text{atan}(ob_x - b_x, ob_y - b_y) - b_d, \quad (9)$$

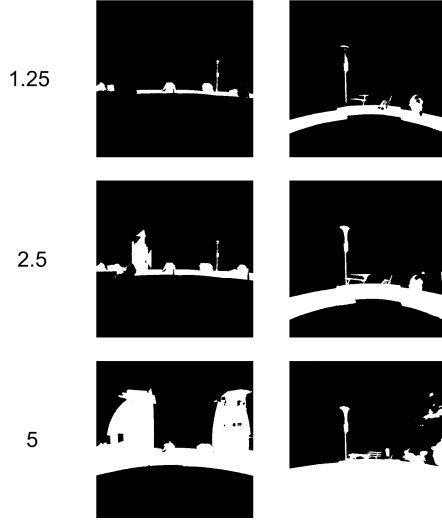


Figure 36: Image showing the effects of selecting different depth ranges O

where b_d is the direction d of the image with regard to north and where ob_x, ob_y, b_x and b_y are the x and y coordinates of the object and image respectively. A range ra is then set on this direction taking into account the distance to the object as they direction should become smaller the further the object is away as the size of the object get smaller i.e.

$$ra_i = di_i * p \quad (10)$$

where p is a variable greater then 0. This range is then applied against the direction of the object i.e.

$$dr_i = (\theta d_i - ra_i, \theta d_i + ra_i) \quad (11)$$

. These two ranges are then combined to generate a mask for every labeled class that is within an image. Figure 36 illustrates the effects of picking different values of o , where we found a value of 2.5 to fit best. Figure 37 illustrates the effects of picking different values for p , where we found that a value of 5 works best as most objects in our set aren't too wide.

The masks are generated for every labeled object within an image. These masks are then combined into a single mask per class i.e. the generated masks for every object are combined into a single mask for each. This mask is then used during training and compared against a normalised CAM.

The masks generated by this method have a size of (512×512) pixels. While this is larger then the CAM that was generated ,i.e. (16×16) , down-scaling the mask to a 16×16 would remove relevant information e.g. see Figure 38. For the purpose of comparing the CAMs with masks, the CAMs are upscled to the size of the masks (512×512) using bilinear upscaling. While this does result in the CAM loss never being zero as the CAM resolution is just too low to perfectly fit the mask, it should still allow for loss to improve the CAMs.

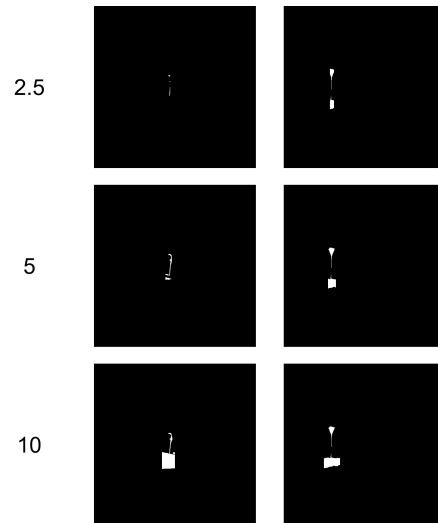
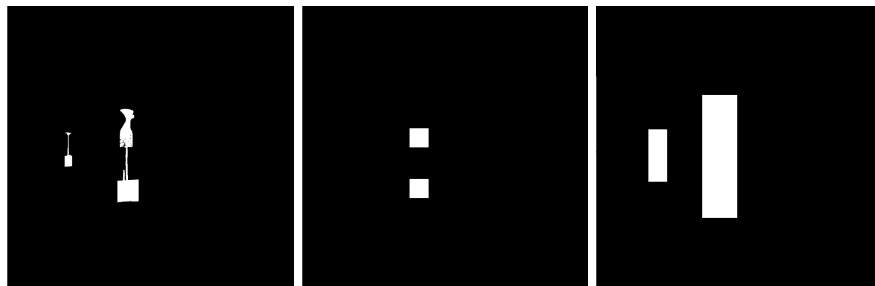


Figure 37: Figure showing the effects of selecting a different width range p



(a) (512×512) depth mask (b) (16×16) depth mask (c) (16×16) depth mask

Figure 38: By down-scaling we lose significant amounts of information (b), if we change how we downscale to instead if any pixel is positive to turn the entire region positive we see a better masks (c). However, the mask (a) can instead be used if we upscale the CAM instead of downscaling the mask

4.3.2 Including CAM Loss Within Training

To see the impact of the interaction between the two different loss functions, the normal cross-entropy loss function and the cam loss function, multiple different combined loss functions were investigated. First l_1 is the normal cross-entropy loss i.e.

$$l_1 = -y \log(\hat{y}) - (1 - y) \log(1 - \hat{y}) \quad (12)$$

where y and \hat{y} are ground truth and the prediction respectively.

Second, l_2 is the CAM loss and consist of a cross-entropy comparison between the CAM and the approximate depth mask i.e.

$$l_2 = -c \log(\hat{c}) - (1 - c) \log(1 - \hat{c}) \quad (13)$$

. Where c and \hat{c} are ground truth (approximate depth mask) and the prediction respectively (CAM). Combining the two losses can then be accomplished in a few ways. We investigated to simply take both losses into account using some weight i.e. resulting in $loss = w_1 l_1 + w_2 l_2$. We also investigated the following relative loss function: $loss = l_1 + l_1 l_2$, to reduce the effects of noise in the dataset. This relative loss function takes the cam loss l_2 more into account when the predicted label l_1 is more wrong. This is desirable as when the loss on the labels is high, the network has a problem selecting the right regions or features for this classification to be made in which case the cam loss can help it focus on the right region. Furthermore, when the loss on the labels is low, the network is apparently already able to find the right features and make correct classification and as such reducing the impact of CAM is desirable as when a good classification we do not want to disturb the region we learned on too much if, for instance, the mask is incorrect due to noise.

4.3.3 Increased Resolution

As our model now has a second form of ground truth that directly gives us a loss on the CAMs, some upscaling techniques can be used to increase the resolution of the CAM. This can help in creating more accurate and expressive CAMs. This is done by adding a binary up scaling layer before the last Resnet residual block increasing the size of the CAM of the last CAM. While the label loss (L1) cannot discriminate between these upscaled values, as it cannot discriminate between the values that came from the same non upscaled value, our depth mask loss (L2) can. This method of upscaling is a technique often used in supervised learning methods for semantic segmentation and bounding box generation. It allows us to increase the resolution of the CAM to at least the resolution of the depth masks. However this upscaling does come at a cost namely the size and memory requirement of the model increase, as these last layers now have an increased size which increases the number of connections. As the memory costs of upscaling the last layers to a (512×512) are too expensive we instead investigate upscaling to a size 32×32 and to a size of 64×64 . Furthermore, these last layers lose their pre-training as they are reconstructed without having pre-trained weights available.

4.3.4 Normalised CAM

As stated before we normalized the CAM within the loss function before comparing them to the automatically generated masks. This is necessary as the values of the CAM are not fixed by a

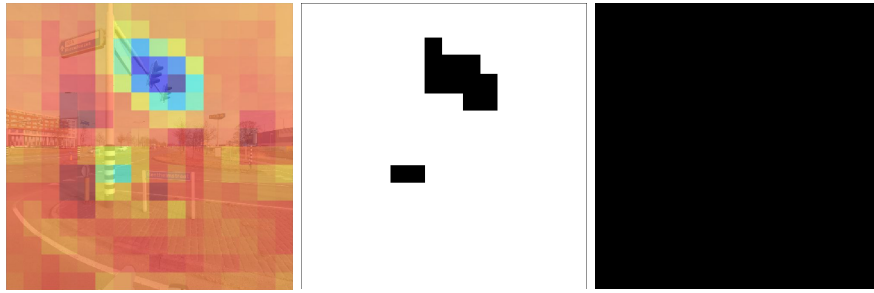


Figure 39: Image A shows a negative CAM i.e. a CAM of a class we predicted not to be within the image. As such due to normalization most of the Image is seen as the peak, resulting in the mask (B) where white pixels are pixels with confidence greater than 0.5. While our mask for this object (C) is empty. This results in a high loss even though the values within the CAM (A) are reasonably low i.e. around 15 which for CAM standards is relatively low. The blue section indicates a large negative part of the CAM this is a result of another object being present within the image that is often not in the same image as the class of this CAM.

certain range, as they are generated based on the activations in the last add or convolutional layer and the weights of the dense layer. Without normalization, we cannot compare the CAM to the automatically generated mask. Normally, the network could learn to have values between a certain range as normalization with Sigmoid or Tanh function could achieve reduce the effectiveness of values past a certain range. However, the CAM is a product of how many features respond to certain class labels which we would not want to change as the number of features that is can differ per class. We, therefore, found the max and minimum values of the CAM and these were mapped to zero and one respectively.

However, when evaluating our result we found two issues with this method. First, by normalizing the CAM, we will always find a region that we consider to be foreground, even if the CAM itself has only low or negative values as illustrated by Figure 39. By normalizing there will always be a max discriminative region even if the values of this region are only slightly higher then the rest of the CAM. This resulted in the CAM loss being very constant even as CAMs improved. As any non-relative changes in the CAM would not have any effect on the loss.

Second, the max or minimum values can massively impact the loss of the entire CAM, resulting in outliers having a major effect on the loss of the CAM.

To solve these problems we investigated multiple methods. First, we investigated using a soft-max normalization, that while would not completely solve these issues it can reduce the effects reduce the effect of negative CAM normalization issue.

Second, we investigate using the confidence of the labels predicted by the model to regulate the normalization process. As the confidence is low for negative CAMs if we multiple the confidence with the normalized with CAM values, the loss generated on these CAM should be majorly reduced. While if the confidence is high the CAM should be selective and accurate. Furthermore, the model is forced to have more confidence for true labels as the loss is higher when having low confidence on a true label as the CAM values are lowered.

Third, we investigate using the average of the top 10% of highest and lowest CAM results when selecting our max and minimum values for normalization this would reduce the effect outliers have on the CAM loss. This third option can also be combined with the first and second suggested improvements.

4.4 Summarize

To summarize we first evaluate the performance of the dataset generation method by manually validating 400 images generated by this method. We evaluate how many labels would be seen as positive for the different label ranges and our proposed mixed label range. After which we train a resnet 50 network on these different label ranges evaluating which gives the best classification performance. We also investigate using multiple different learning rates to see what kind of effect this has on the noise labels we are training on. The best label range model is then used as baseline to compare the effects of using CAM loss against. This is done on a image-wide label level, a CAM intersection level, bounding box level and finally on a Semantic segmentation level. While good results for semantic segmentation our the key goals, getting accurate results on a image-wide or CAM intersection level would already provide Cyclomedia with useful information.

The CAM loss models we will be investigated can not only be split up by their specific loss function but also by what kind of layer the use to generate the CAM. Either the add layer or the last convolutional layer. For all CAM loss models we investigate both. We also investigate using different learning rates for all these models. Finally, we investigate different cam normalization techniques. This results in the following list of models see Table 4.

Finally, we also train two additional CAM loss model with increased resolution were we select the best performing model. Which, we then upscale to 32×32 and upscale to 64×64 .

Loss name	Loss function	CAM layer	Learning rate	CAM normalization
Relative loss	$L_1 + L_1L_2$	add	0.001	Normal
Relative loss	$L_1 + L_1L_2$	conv	0.001	Normal
Relative loss	$L_1 + L_1L_2$	add	0.001	Normal
Relative loss	$L_1 + L_1L_2$	add	0.01	Normal \times Confidence
Related loss	$L_1 + L_1L_2$	add	0.01	softmax
Equal loss	$0.5L_1 + 0.5L_2$	add	0.001	Normal
Equal loss	$0.5L_1 + 0.5L_2$	conv	0.001	Normal
Equal loss	$0.5L_1 + 0.5L_2$	add	0.01	Normal
Equal loss	$0.5L_1 + 0.5L_2$	add	0.01	Normal \times Confidence
Equal weighted loss	$0.5L_1 + 0.5L_2$	add	0.01	Softmax
Label less loss	$0.3L_1 + 0.7L_2$	add	0.001	Normal
Label less loss	$0.3L_1 + 0.7L_2$	conv	0.001	Normal
Label less loss	$0.3L_1 + 0.7L_2$	add	0.01	Normal
Label less loss	$0.3L_1 + 0.7L_2$	add	0.01	Normal \times Confidence
Label less loss	$0.3L_1 + 0.7L_2$	add	0.01	Softmax
Depth less loss	$0.7L_1 + 0.3L_2$	add	0.001	Normal
Depth less loss	$0.7L_1 + 0.3L_2$	conv	0.001	Normal
Depth less loss	$0.7L_1 + 0.3L_2$	add	0.01	Normal
Depth less loss	$0.7L_1 + 0.3L_2$	add	0.01	Normal \times Confidence
Depth less loss	$0.7L_1 + 0.3L_2$	add	0.01	Softmax

Table 4: Different CAM loss models

5 Results

In this section we show the result acquired on our experiments. First we go over the metrics used to evaluate the experiments in section 5.1. After which we show the in order the result of the following experiments. First in section 5.2 we evaluate the automatically generated dataset. After which we show the results acquired training on this generated dataset in section 5.3. Then we show our result acquired using our CAM loss model in section 5.4. Finally, followed by the results acquired using Upscaled CAM loss models in section 5.5.

5.1 Metrics

To evaluate the performance of our dataset, we will need some metrics. This section will go over commonly used metrics for classification purposes and the metrics used for evaluating shapes such as bounding boxes and masks. To do this, we need to distinguish the four types of predictions our method can make. A correct prediction in which the method generated label is positive and the validation set has the same positive label i.e. a True Positive (TP). A correct prediction in which the method generated label is negative and the validation set has the same negative label i.e. a True Negative (TN). An incorrect prediction in which the method generated label is positive and the validation set instead has a negative label i.e. a False Positive (FP). Finally, there is the incorrect prediction in which the method-generated label is negative and the validation set instead has a positive label i.e. a False Negative (FN).

With these four different types of predictions, we evaluate the model in three different ways: recall, precision, and F-score

The recall metrics can inform us about how many of the labels that exist we found and is calculated as such

$$\text{Recall} = \text{TP}/(\text{TP} + \text{FN}). \tag{14}$$

The precision metric can inform us about how many of the labels we found are correct and is calculate as such

$$\text{Precision} = \text{TP}/(\text{TP} + \text{FP}). \tag{15}$$

Recall and precision are key metrics in understanding how a model performs. However, looking at each separately will give an incomplete picture of the performance a method might have. For example, when evaluating a model by only looking at recall, we could be misled as a model might just always output a true label in every image. This would result in a perfect recall score of 1 as FPs are not accounted for when computing recall. The same could happen with precision when for instance the method only makes one positive output while missing all other objects and this positive output is correct, it would result in a perfect precision score of 1.

What is needed is a metric that combines the recall and precision. An often-used metric for combining recall and precision is called the F-score. The F-score metric simple tries to take into account both recall en precision i.e.

$$\text{Fscore} = 2 \times ((\text{precision} \times \text{recall})/(\text{precision} + \text{recall})). \tag{16}$$

The F-score can be used to give a general idea of the performance of a model.

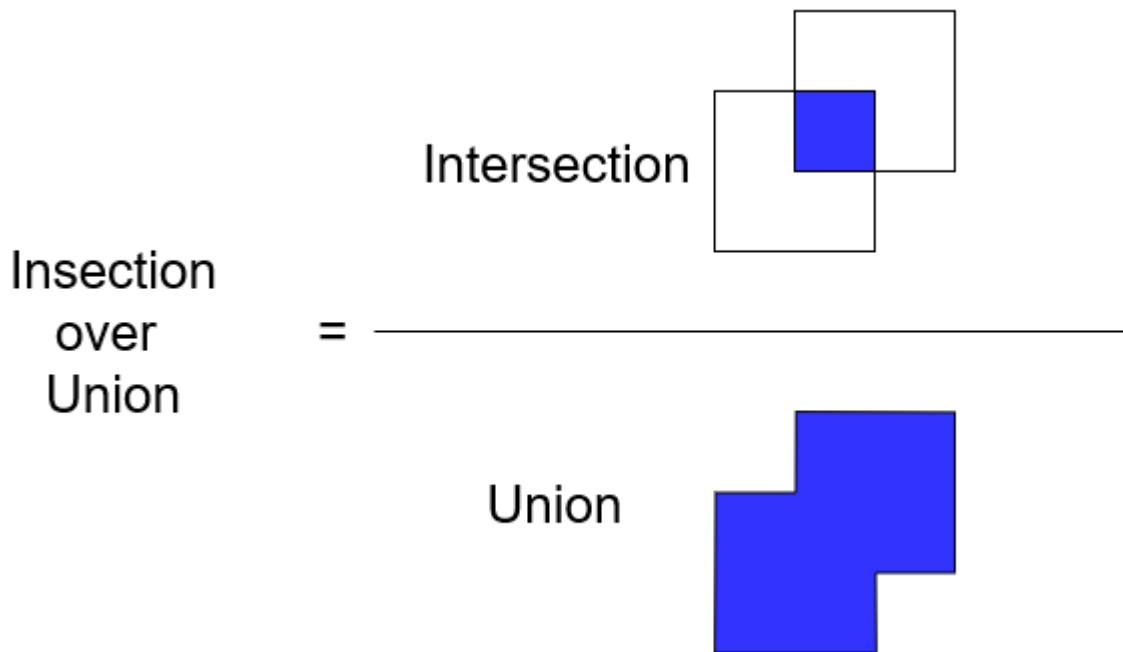


Figure 40: Intersection over Union is calculated between two shapes by dividing the area of union of the two shapes by area of their intersection.

While we now have some metrics for evaluating image-wide predictions, these metrics are insufficient to evaluate shapes such as CAMs, bounding boxes, and masks. For these shapes, the definition when a good classification is made needs to be changed. Unlike image-wide object detection, all the individual objects/regions of a class that are visible need to be found. Furthermore, an additional metric is needed that can evaluate how well we accomplished this. With this additional metric, a judgment can be made if sufficient of the objects/object were found for a good classification (TP) to have been made. A commonly used metric for this is IOU (Intersection over Union). IOU can find how identical two shapes are in both position and shape in a singular metric. To calculate the IOU of two shapes the area of the intersection between both shapes is taken and divided by the area of the union union of both shapes as illustrated in Figure 40. This results in a number between (0,1) that states how much the shapes are identical. An example of possible IOU scores is shown in Figure 41.

While the IOU can give a great insights in how well two shapes are related. This metrics is not perfect for judging the quality of CAMs. A cam is not focused on a perfect IOU but on recognizing some key features of the objects. Towards this goal, we decide to introduce an additional metric that directly judge the CAMs generated that doesn't rely on IOU. The goals of this metric is to investigate how well the CAM are able to localize all of the objects. Where we only want to know if the highest points of the CAMs intersect with the semantic segmentation groundtruth. Counting it positive if it does intersect. As we take the highest points of the CAMs this should not preference CAMs that fill the entire screen as these will not have any peaks. This metrics allows us to quickly

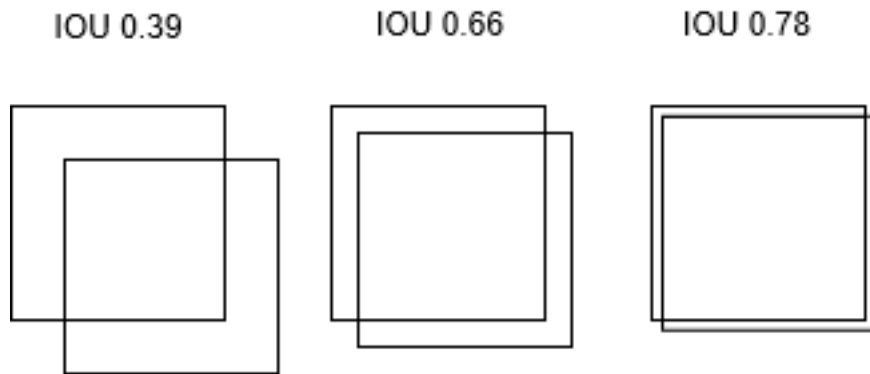


Figure 41: Examples of the IOU being calculated on two shapes.

see if our models are able to find at least a part of the object.

To evaluate the predictions made by our models, a ground-truth set is needed. For this, we manually annotated 250 images with highly accurate polygons. An example of these annotations is shown in Figure 42. These polygons can then be converted into segmentation masks, when evaluating our semantic segmentation results and our CAM intersection results, and converted into bounding boxes when evaluating our bounding box results.



Figure 42: Example of manually annotated images using polygons.

5.2 Evaluating Automatic Dataset Generation Method

In this section we evaluate our automatically generated dataset as described by our method in section 4.1.

The automatic dataset generation method generated a dataset of 20,000 images, 1000 images for each class. This set was then annotated with labels up to 50 meters, allowing us to easily investigate values for our label range X_{min} and X_{max} (see Table 5). The images were then split up into a training set of 16,000 images and a validation set of 4,000 images.

Objects	5	10	15	20	25	30	35	40	45	50
Traffic light	+332	+800	+156	+84	+60	+64	+39	+38	+12	+7
Barrier post	+775	+751	+205	+152	+141	+146	+140	+104	+49	+22
Utility cabinet	+171	+1184	+398	+308	+269	+258	+283	+143	+60	+17
Traffic sign	+525	+1606	+648	+492	+449	+386	+307	+213	+93	+37
Manhole	+723	+1314	+935	+826	+718	+700	+647	+391	+181	+49
Barrier/Gate	+204	+650	+28	+13	+11	+18	+8	+7	+5	+0
Street gully	+1368	+2440	+1751	+1215	+826	+609	+438	+268	+122	+43
Electronic freeway sign	+430	+558	+64	+41	+32	+25	+16	+16	+12	+3
Bicycle stand	+290	+601	+35	+38	+42	+47	+37	+19	+14	+5
Street name sign	+197	+749	+58	+56	+49	+33	+50	+28	+8	+4
Garbage Can	+259	+1047	+263	+173	+169	+152	+137	+85	+29	+14
Street light	+860	+2664	+1896	+1879	+1521	+1170	+816	+382	+159	+48
Camera	+120	+702	+22	+32	+29	+33	+27	+25	+3	+2
Bench	+248	+836	+141	+140	+109	+102	+94	+59	+25	+12
Roadside protection block	+355	+488	+22	+24	+15	+14	+16	+8	+7	+0
Information sign	+167	+683	+28	+48	+33	+32	+33	+22	+8	+5
Public play equipment	+114	+747	+99	+50	+28	+39	+27	+18	+14	+6
Hectometer post	+138	+926	+171	+152	+157	+225	+227	+141	+61	+17
Ladder	+79	+762	+44	+24	+27	+25	+36	+24	+17	+4
Roadsign direction Arrow	+277	+586	+31	+36	+72	+94	+91	+48	+25	+7

Table 5: Number of positive images added per class by increasing the label range X_{min} .

To evaluate the quality of the generated set, we compared it to our manually annotated labels of the same images. Doing so reveals a clear difference between our manually annotated labels, i.e. ground truth, and the method provided labels as illustrated in Figure 43. It should be noted that we are evaluating on an image-wide level and as such even any distinguishability between the number of objects for a given class is lost. For example, if a particular image contains twenty traffic signs, it would still receive the same labels as an image only containing one traffic sign.

This difference is largely dependent on the label range per class. However, even when selecting the best possible range for the objects, a large discrepancy can be seen between the ground truth and method generated labels. Table 6 shows that this difference seems to be the largest for smaller objects e.g. cameras, manholes, and hectometer posts.

The selecting of objects based on size and visibility in the road environment seems relatively successful as the performance seems to be close or on the best performing range for most objects and certainly better than selecting a generic range for all objects. We can also see that increasing the label range for some of the smaller objects does not decrease the performance as fast as we would expect. For example, street name sign performance does not drop as much after a label range of 25 meters, while these objects are certainly not visible in most cases after 25 meters. This is explained when we look at how many positive images for a certain class get added at the larger ranges as shown in Table 5. For every range increase, the total number of negative images is decreased resulting in a smaller pool of images being available to be added to the larger label range. This massively reduced the number of added, positively-labeled images per class per increase of range. Furthermore, after a range of 10 meters, a massive decrease for most objects can be seen. This is due to the fact that the 800 images per class are found within the 3-10 meter range. After which an increase can only be found when an object is within the background of another object. For certain classes that occur very frequently within the road environment this happens more often as we see with street lights, street gullies, and manholes. It is also related to how often an object is located near another object. For instance, traffic lights occur often together with traffic signs and as such traffic signs are often within the images of traffic lights.

The advantage of adding hard labels can be seen in Table 7, where we are able to reduce the number of false negatives at the cost of some of the true negatives. As the number of true negatives is much larger than the number of false negatives, we suppose that this trade-off is worth it when training on the dataset. In the comparison against the manual validation set, true negatives do not have any effect on the F-score, as such the hard label improves the overall results. While this performance increase is small, the performance increase is bigger for classes that were underestimated in terms of best visible range e.g. see street gully.

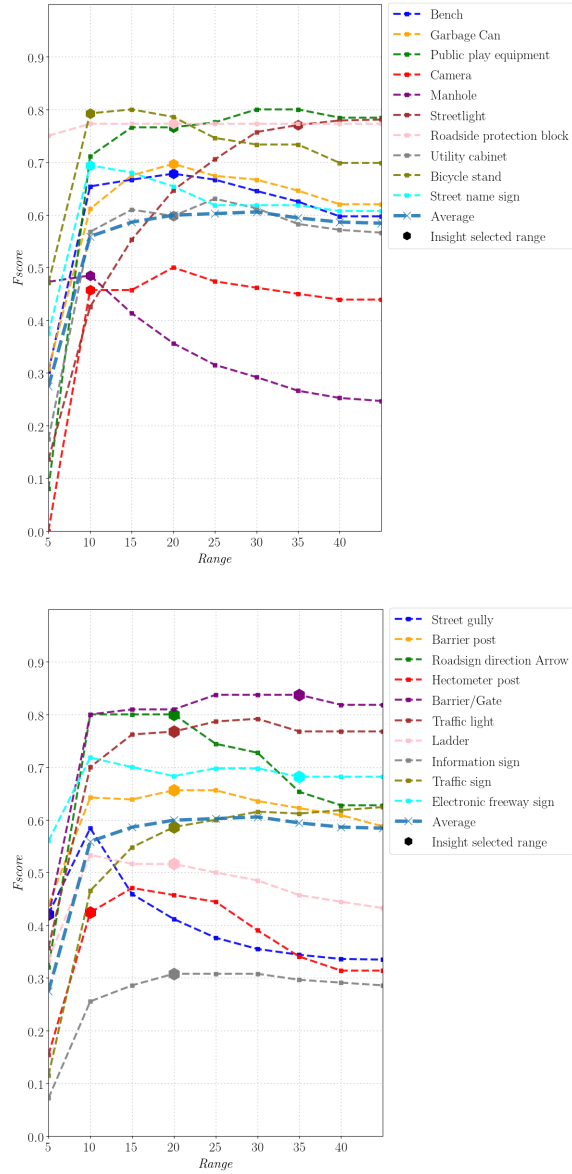


Figure 43: F-score for the 20 classes at different label X_{min} ranges

	True	False
positive	18	7
negative	375	0

(a) Barrier/gate 0.84

	True	False
positive	17	3
negative	373	7

(d) Roadside protection block 0.77

	True	False
positive	18	5
negative	371	6

(g) Public play equipment 0.77

	True	False
positive	15	12
negative	371	2

(j) Electronic freeway sign 0.68

	True	False
positive	26	20
negative	339	15

(m) Utility cabinet 0.60

	True	False
positive	24	47
negative	325	4

(p) Manhole 0.48

	True	False
positive	16	6
negative	340	38

(s) Street gully 0.42

	True	False
positive	16	4
negative	376	4

(b) Roadsign direction arrow 0.8

	True	False
positive	203	86
negative	76	35

(e) Street light 0.77

	True	False
positive	21	7
negative	368	4

(c) Bicycle stand 0.79

	True	False
positive	33	7
negative	347	13

(f) Traffic light 0.77

	True	False
positive	31	15
negative	342	12

(h) Garbage can 0.72

	True	False
positive	17	7
negative	368	8

(i) Street name sign 0.69

	True	False
positive	20	14
negative	361	5

(k) Bench 0.68

	True	False
positive	41	30
negative	316	13

(l) Barrier post 0.66

	True	False
positive	8	15
negative	377	0

(o) Ladder 0.52

	True	False
positive	8	12
negative	373	7

(q) Camera 0.46

	True	False
positive	7	17
negative	374	2

(r) Hectometer post 0.42

	True	False
positive	8	18
negative	356	18

(t) Information sign 0.31

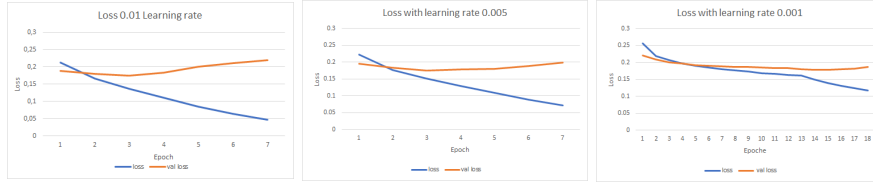
Table 6: The mixed range label results on the manual validation set with F-scores.

Objects	Correctly removed FPs	Falsely removed TNs
Traffic light	5	5
Barrier post	0	1
Utility cabinet	4	7
Traffic sign	8	7
Manhole	1	21
Barrier/Gate	0	4
Street gully	24	37
Electronic freeway sign	0	0
Bicycle stand	1	1
Street name sign	0	1
Garbage Can	0	4
Street light	7	4
Camera	0	0
Bench	0	3
Roadside protection block	0	0
Information sign	0	0
Public play equipment	2	1
Hectometer post	1	0
Ladder	0	2
Roadsign direction Arrow	0	4
Sum	53	102

Table 7: The number of correctly removed FPs and Falsely removed TNs on the validation set consisting of 400 images labels with a mixed-label range.

	True	False		True	False		True	False
positive	33	7	positive	18	5	positive	21	7
negative	342	8	negative	370	4	negative	369	3
(a) Traffic light 0.81 (+0.04)			(b) Public play equipment 0.8 (+0.03)			(c) Bicycle stand 0.81 (+0.02)		
	True	False		True	False		True	False
positive	203	86	positive	26	20	positive	16	6
negative	72	28	negative	332	11	negative	303	14
(d) Street light 0.78 (+0.01)			(e) Utility cabinet 0.63 (+0.03)			(f) Street gully 0.62 (+0.20)		
	True	False		True	False		True	False
positive	68	19	positive	24	47	positive	7	17
negative	229	69	negative	304	3	negative	374	1
(g) Traffic sign 0.61 (+0.02)			(h) Manhole 0.49 (+0.01)			(i) Hectometer post 0.44 (+0.02)		

Table 8: The mixed range label results on the manual validation set with false negative and true negative removed as indicated by Table 7.



(a) Epoch three acquired lowest validation loss with a value of 0.175 (b) Epoch three acquired lowest validation loss with a value of 0.174 (c) Epoch 15 acquired lowest validation loss with a value of 0.179

Figure 44: Resnet 50 mixed models loss with different learning rates.

5.3 Evaluating The Model Trained On The Automatically Generate Dataset

In this section, we evaluate the performance of the mixed label-range Resnet 50 model. First, we evaluate the result on a image-wide level. Followed by evaluating the model on the quality of the CAM generated. Finally, we investigate the performance of bounding boxes and semantic segmentation masks generated on the CAMs.

5.3.1 Classification

With our mixed label-range, we trained three models all with different learning rates i.e. 0.01, 0.005 and 0.001. Figure 44 illustrates the loss we acquired while training on these different learning rates. We see that the model with the higher learning rate, i.e. 0.01, validations loss converges very fast before increasing again. This usually means that the model has a too high learning rate. The medium learning rate, i.e. 0.005 achieves the lowest validation loss out of the three models, which we found unexpected as the lower learning rate should allow it to converge to a lower value. Even more unexpected is that the low learning rate, i.e. 0.001, validation loss convergence at the highest point, while we would expect smaller steps in the low learning rate to provide more fine-tuning.

To measure the performance of the method, we validate them against a manually image-wide dataset consisting of 400 images. We do not use the automatically constructed validation set as this set can give some misleading results. We found this when investigating our classification results on the different label ranges. Table 9 illustrates these misleading results we can obtained on different label ranges. Namely, we found that the performance of certain smaller objects, i.e. street gullies and manholes, would increase when increasing the label range. This was unexpected as at the larger label ranges the objects would never be visible.

In Figure 45, we show some of the CAMs of a smaller class that should not be visible at large ranges, and it becomes clear why the performance increased. Instead of learning the appearance of these smaller objects, the network learned the environment these objects are usually situated in. Now, when the label range increases, the model is pushed into learning the environment as the object itself is too far away to be visible. When we now further increase the label range, the performance is also increased as the chance of the image being labeled correctly that a certain object is in that environment increases. The automatically generated validation set shares these faults with the automatically generated training set which results in the model having increased

Objects	Mixed range	10	20	30	40
Average	0.69	-0.06	-0.04	-0.02	-0.01
Traffic light	0.79	-0.02	+0.01	-0.04	-0.0
Barrier post	0.46	+0.09	+0.05	+0.04	+0.07
Utility cabinet	0.52	-0.03	-0.01	0.0	-0.0
Traffic sign	0.56	-0.06	+0.04	-0.0	-0.02
Manhole	0.38	+0.05	+0.15	+0.2	+0.23
Barrier/Gate	0.83	+0.03	+0.01	-0.02	0.0
Street gully	0.33	+0.2	+0.36	+0.41	+0.44
Electronic freeway sign	0.8	+0.02	+0.02	-0.03	-0.01
Bicycle stand	0.75	-0.02	-0.02	-0.14	-0.07
Street name sign	0.51	-0.05	-0.03	-0.03	-0.2
Garbage Can	0.55	0.0	-0.05	-0.08	-0.03
Street light	0.86	-0.17	-0.1	-0.02	0.0
Camera	0.59	-0.01	-0.01	-0.03	-0.07
Bench	0.5	0.0	+0.07	-0.05	+0.01
Roadside protection block	0.65	+0.03	-0.01	-0.06	-0.07
Information sign	0.37	+0.06	+0.07	+0.01	-0.06
Public play equipment	0.73	-0.0	-0.0	-0.06	-0.12
Hectometer post	0.69	+0.01	-0.02	-0.16	-0.12
Ladder	0.74	0.0	-0.0	-0.03	-0.09
Roadsign direction Arrow	0.72	+0.1	+0.01	-0.04	-0.15

Table 9: Performance when training on different values for X_{min} , where $X_{max} = X_{min} + 10$. Evaluation on the automatically generated validation set.

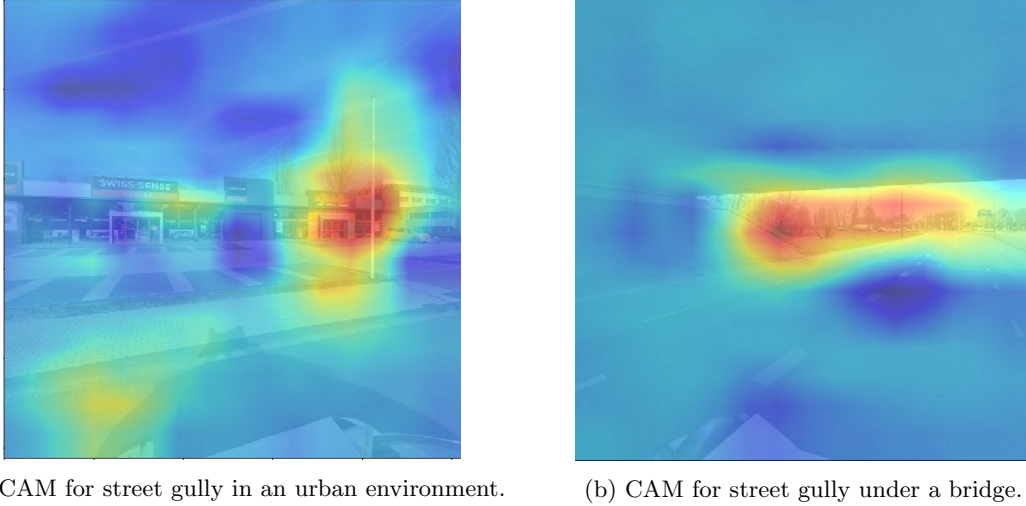


Figure 45: CAMs of street gullies showing that instead of detecting the street gully it detects the environment it is in.

performance on this validation set. This problem could also be solved by specifically adding negative images for certain environments, however this is almost impossible for certain objects. For instance, within an urban living environment a street gully is located almost every few meters. Normally, when manually annotating this problem can be avoided as many street gully are blocked or by annotating bounding boxes resulting in the rest of the images functioning as a negative images. However, we do not have access to either, we therefore do not specifically look for negative images in any environment and instead use the background of other objects as negatives.

As such we use the manually constructed validation set for validating the performance. Table 10 shows the results acquired on this validation set, exposing the problems with having a larger label range. Nevertheless, some of the less common objects do not seem to gain or lose in performance. This confirms our results in Section 5.2 where we concluded that for these scarcer objects only a few labels get added at the larger label range, and the network is able to largely ignore these noisy labels.

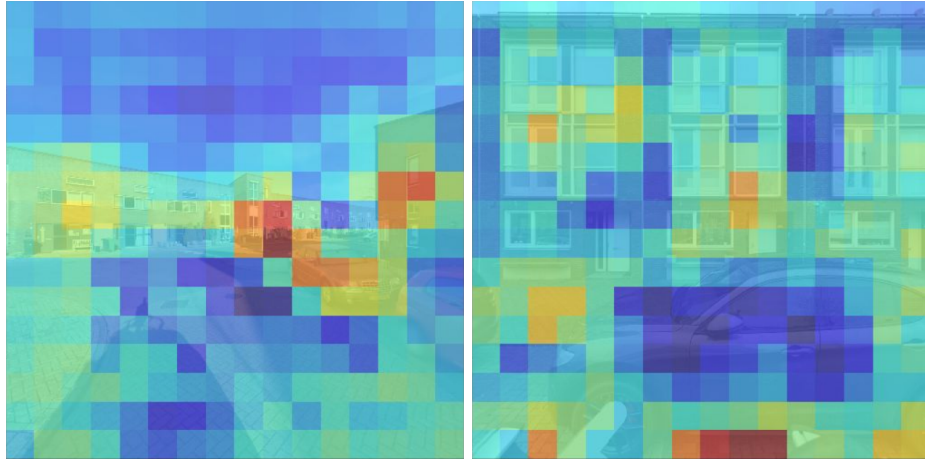
Table 11 shows the result acquired using the different learning rate models on the manually created validation set. We see that the higher learning rate model is the best performing network compared to the lower and medium learning rate models. To understand why this happens we look at the CAMs of the three networks. The CAMs, as shown in Figure 46, can shows that the model with lower learning rates starts to take more of the background features into account leading it to worse performance against the manual validation set. This would also mean that the more erroneously labeled classes should perform better with the lower learning rate, which for some of the classes we also see i.e. manhole, street gully, hectometer post. Interestingly, we do not see it for Information Sign but this can be explained when taking into account that information sign is a class consisting of many different sub-classes with widely different shapes. It thus takes the network longer to learn this object.

Objects	Mixed range	10	20	30	40
Average	0.64	-0.15	-0.06	-0.07	-0.09
Traffic light	0.74	-0.12	-0.02	-0.05	+0.03
Barrier post	0.43	+0.1	+0.1	+0.08	+0.12
Utility cabinet	0.62	0.0	+0.1	-0.01	-0.07
Traffic sign	0.51	-0.18	+0.14	+0.01	-0.01
Manhole	0.52	+0.09	-0.24	-0.24	-0.27
Barrier/Gate	0.92	+0.02	-0.03	-0.14	+0.02
Street gully	0.28	+0.05	+0.0	+0.06	+0.02
Electronic freeway sign	0.89	-0.03	-0.08	+0.05	+0.03
Bicycle stand	0.88	-0.07	-0.06	-0.18	-0.22
Street name sign	0.63	+0.04	-0.02	-0.13	-0.14
Garbage Can	0.52	-0.15	-0.03	+0.01	-0.03
Street light	0.75	-0.36	-0.07	-0.0	-0.02
Camera	0.42	+0.16	+0.12	+0.17	+0.08
Bench	0.51	-0.14	+0.06	0.0	+0.09
Roadside protection block	0.65	+0.1	+0.11	+0.02	-0.04
Information sign	0.3	-0.02	-0.06	-0.03	-0.15
Public play equipment	0.79	-0.1	-0.0	-0.08	+0.04
Hectometer post	0.54	-0.1	-0.1	+0.17	-0.13
Ladder	0.57	-0.03	-0.05	+0.06	-0.14
Roadsign direction Arrow	0.73	+0.04	0.0	-0.0	-0.15

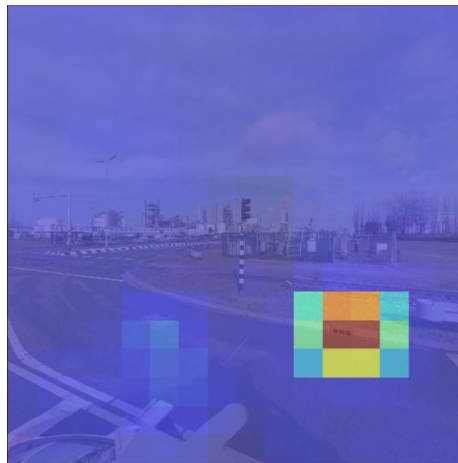
Table 10: F-scores for the different label range models on the manually created validation set.

Learning rate	0.01	0.005	0.001
average	0.64	0.63	0.62
Barrier post	0.43	+0.08	+0.09
Garbage Can	0.52	-0.05	-0.02
Bench	0.51	+0.14	-0.02
Roadside protection bloc	0.65	+0.12	-0.03
Bicycle stand	0.88	-0.08	-0.06
Hectometer post	0.54	-0.06	-0.1
Information sign	0.3	-0.01	-0.03
Manhole	0.52	-0.02	-0.16
Utility cabinet	0.62	0.0	-0.01
Street gully	0.28	-0.19	-0.16
Street light	0.75	+0.01	-0.0
Electronic freeway sign	0.89	-0.02	-0.06
Roadsign direction Arrow	0.73	+0.11	+0.05
Barrier/Gate	0.92	0.0	0.0
Public play equipment	0.79	-0.0	-0.03
Street name sign	0.63	-0.07	-0.04
Ladder	0.57	-0.07	-0.0
Camera	0.42	-0.05	+0.15
Traffic sign	0.51	-0.05	+0.03
Traffic light	0.74	0.0	-0.05

Table 11: Difference in F-score when using a different learning rate on manual validation set.



(a) Low learning rate learning the environment resulting in this FP CAM. (b) Medium learning rate learning the environment resulting in this FP CAM.



(c) High learning rate learning only to object resulting in this TP detection with accurate CAM.

Figure 46: CAMs for the street-gully object at different learning rates.

Learning rate	0.01		0.001		0.005	
CAM generating layer	Add	Conv	Add	Conv	Add	Conv
Average	0.37	-0.0	0.33	-0.05	0.36	-0.01
Traffic light	0.36	+0.01	0.35	+0.03	0.4	-0.03
Barrier post	0.24	+0.01	0.35	-0.08	0.28	-0.02
Utility cabinet	0.32	0.0	0.22	-0.15	0.25	0.0
Traffic sign	0.4	-0.04	0.29	-0.05	0.27	-0.0
Manhole	0.38	-0.02	0.25	+0.06	0.3	+0.03
Barrier/Gate	0.76	0.0	0.73	+0.03	0.75	+0.01
Street gully	0.27	0.0	0.0	0.0	0.04	-0.04
Electronic freeway sign	0.32	+0.05	0.22	+0.02	0.35	-0.01
Bicycle stand	0.25	0.0	0.22	-0.06	0.0	0.0
Street name sign	0.0	0.0	0.07	-0.07	0.0	0.0
Garbage Can	0.51	-0.05	0.48	-0.04	0.59	-0.04
Street light	0.5	-0.04	0.44	0.0	0.48	-0.05
Camera	0.18	0.0	0.14	+0.12	0.2	+0.01
Bench	0.57	0.0	0.57	-0.07	0.5	0.0
Roadside protection block	0.43	-0.02	0.44	-0.28	0.34	0.0
Information sign	0.15	0.0	0.09	0.0	0.06	+0.0
Public play equipment	0.22	+0.01	0.29	+0.04	0.73	0.0
Hectometer post	0.61	0.0	0.5	-0.28	0.6	0.0
Ladder	0.5	+0.06	0.67	-0.33	0.67	+0.1
Roadsign direction Arrow	0.41	-0.01	0.3	+0.07	0.38	-0.08

Table 12: CAM intersection F-scores of the non CAM loss models, with different learning rates and CAM generation using the last 'add' layer and the last convolutional layer.

5.3.2 CAM Performance

In this section, we investigate the performance of the mixed label Resnet 50 model on the CAMs generated by the networks using the intersection metric as explained in Section 5.1.

Table 35 shows the localizational ability of our CAMs, generated by the Resnet network, using the intersection metric. We can see an overall poor performance when setting our threshold at reasonable levels i.e. only taking the top 20%. It also again shows that a high learning rate is best and that CAM generation produces the best results by using the 'add' layer to generated CAM instead of using the last convolutional layer. The difference we describe in section 4.2.2.

While some CAMs can be considered accurate, Figure 47 shows some of the issues with the CAMs. First, they seem to have trouble finding multiple objects of the same class, instead of fixating on a single object per class. The cases where the CAMs do find multiple objects are usually the cases were these objects are right next to each other which results in slightly larger CAM in that region. Second, for certain situations, the CAMs simply focus on the most common place where a specific object would locate e.g. a ladder is often located at a roadside barrier. Thirds, another issue is when two objects are often near each other, as in the case of utility cabinets and electronic freeway signs, they often get confused with each other. This results in some cases to a large number of

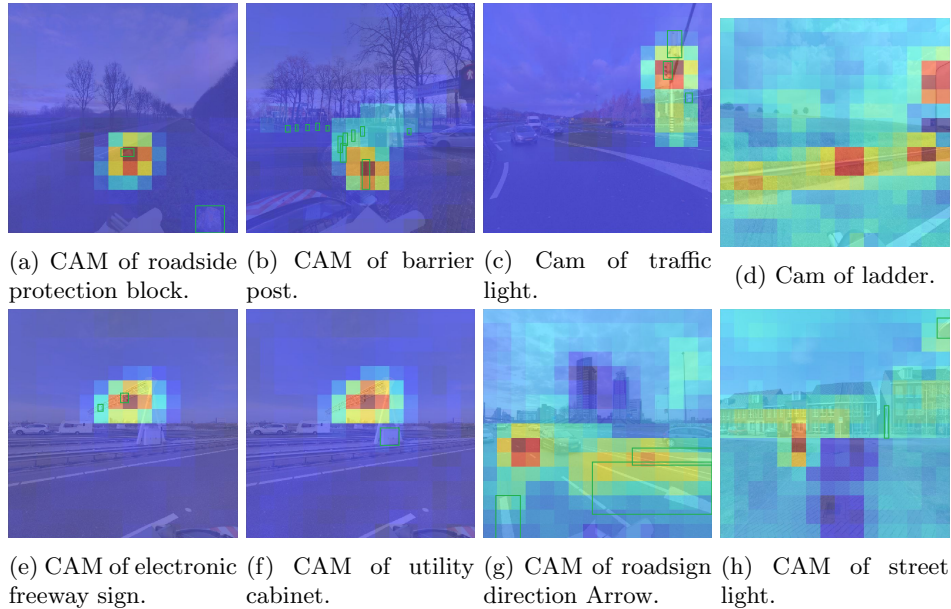


Figure 47: Common faults with CAMs. While (a)(b)(c) have correct CAM for a single object the miss additional classes which are also present within the environment. Image (d) showing the CAM picking the most commonplace where the object is often present. Furthermore, it shows an activation on another object which is also often present within its environment. Image (e) and (f) show the CAM of two objects that are often present within the same image, which in this case results in them learning to respond to the same features. Image (g) and (h) show the cam missing the correct objects.

features overlapping with each other and becoming indistinguishable. Resulting in almost identical CAMs that are not situated on the correct objects.

5.3.3 Bounding Box And Semantic Segmentation Performance

In this section, we illustrate the performance of the bounding boxes and semantic mask generated by the best performing Resnet model i.e. the mixed range model with a high learning rate of 0.01 and CAM generation using the last 'add' layer.

Table 14 illustrates the performance of the model using the two CAM to bounding box methods as outlined in section 4.2.3 with multiple different threshold values. Where the region proposal method has increased performance at the higher IOU ranges while being less accurate at the lower IOU ranges. This seems logical as the region proposal will sometimes already be able to find the object and bound it. The CAM then just has to intersect with the object resulting in a high IOU score. However, this approach is highly dependent on the generated region proposal and on what this region proposal algorithm decides to focus on. This result in it falling more often especially for certain object.

Object	0.0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
Traffic light	0.63	0.26	0.12	0.09	0.03	0.0	0.0	0.0	0.0	0.0
Barrier post	0.67	0.11	0.02	0.02	0.0	0.0	0.0	0.0	0.0	0.0
Utility cabinet	0.5	0.27	0.18	0.09	0.0	0.0	0.0	0.0	0.0	0.0
Traffic sign	0.62	0.13	0.05	0.01	0.0	0.0	0.0	0.0	0.0	0.0
Manhole	0.5	0.22	0.17	0.17	0.09	0.0	0.0	0.0	0.0	0.0
Barrier/Gate	0.93	0.56	0.19	0.0	0.0	0.0	0.0	0.0	0.0	0.0
Street gully	0.27	0.18	0.09	0.09	0.09	0.05	0.05	0.0	0.0	0.0
Electronic freeway sign	0.8	0.13	0.04	0.04	0.0	0.0	0.0	0.0	0.0	0.0
Bicycle stand	0.25	0.25	0.25	0.25	0.25	0.25	0.0	0.0	0.0	0.0
Street name sign	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
Garbage Can	0.7	0.39	0.32	0.25	0.07	0.07	0.07	0.0	0.0	0.0
Street light	0.77	0.4	0.18	0.05	0.01	0.01	0.0	0.0	0.0	0.0
Camera	0.36	0.09	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
Bench	0.75	0.5	0.43	0.39	0.25	0.11	0.11	0.04	0.0	0.0
Roadside protection block	0.77	0.16	0.1	0.07	0.03	0.0	0.0	0.0	0.0	0.0
Information sign	0.15	0.15	0.15	0.15	0.08	0.08	0.0	0.0	0.0	0.0
Public play equipment	0.34	0.23	0.21	0.11	0.0	0.0	0.0	0.0	0.0	0.0
Hectometer post	0.7	0.35	0.17	0.0	0.0	0.0	0.0	0.0	0.0	0.0
Ladder	0.5	0.5	0.4	0.2	0.2	0.0	0.0	0.0	0.0	0.0
Roadsign direction arrow	0.75	0.15	0.1	0.05	0.05	0.0	0.0	0.0	0.0	0.0

Table 13: Performance decrease when increasing the IOU for bounding box generation with region proposals. It also show that certain objects with high performance drop faster then others. For example, Barrier post, Traffic sign, Electronic freeway sign, Camera, RoadSide protection block, Roadsign direction arrow.

The simple bounding box creation method can also illustrate the difficulty of correctly mapping the entire object. As CAMs are known for only activating on the most discriminative/selective parts of an image, objects that do not fit the resolution of the CAM can have increased in difficulties as the CAM cannot adequately describe them. Table 13 illustrates that this issue is also present with our model. This is best illustrated by looking at the performance lost when increasing the minimum IOU for a positive classification to be made. If we look at the difference in performance by increase the IOU threshold from 0.1 to 0.2, we can clearly see certain objects drop faster in performance then others. Objects like traffic lights, barrier posts, traffic signs, barriers/gates, street gullies, electronic freeway signs, street lights, cameras and hectometer posts all drop significantly. All these objects cannot be represented by the CAM very accurately due to there size being either too small or too big, which is illustrated by Figure 48. Finally, objects like electronic freeway signs and traffic lights can have an additional problem as these objects are rarely alone within the image, usually even situated together. This results in larger merged CAMs that our method cannot separate into two different bounding boxes or masks leading to a bounding box or mask too large to adequately fit the objects.

Table 15 shows the performance of the model using the two semantic-segmentation methods. We can see an increase in performance over the bounding box method while normally one would expect semantic segmentation to be a harder problem. However, our generated masks are better at

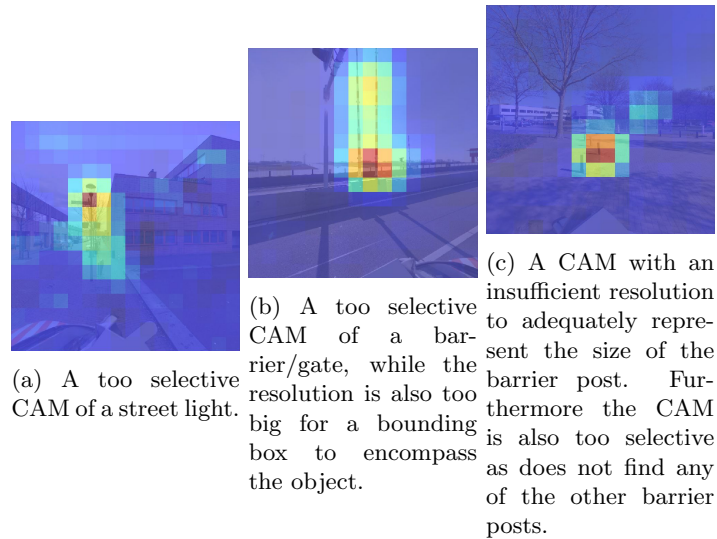


Figure 48: Examples of too selective CAMs or too big CAMs due to the size of the object and the resolution of the CAM.

thresholds	Without region proposals			With region proposals				
Foreground threshold	30%	20%	10%	30%	20%	10%	10%	10%
Background threshold	-	-	-	40%	40%	40%	50%	30%
0.0	0.55	0.55	0.55	0.55	0.55	0.55	0.55	0.55
0.1	0.25	0.25	0.22	0.18	0.15	0.17	0.18	0.17
0.2	0.13	0.16	0.14	0.1	0.1	0.11	0.09	0.11
0.3	0.09	0.1	0.08	0.07	0.08	0.08	0.07	0.09
0.4	0.03	0.06	0.05	0.06	0.06	0.07	0.05	0.08
0.5	0.01	0.03	0.02	0.04	0.03	0.04	0.04	0.05
0.6	0.0	0.01	0.0	0.03	0.02	0.03	0.03	0.03
0.7	0.0	0.0	0.0	0.02	0.01	0.02	0.01	0.02
0.8	0.0	0.0	0.0	0.0	0.01	0.0	0.0	0.01
0.9	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0

Table 14: Table showing our CAM to bounding box generation methods results in F-scores for multiple thresholds. While the bounding box method without region proposals on average generated the best results. At higher IOUs the addition of region proposals produces better results.

Method	Thresholded CAM			GrabCut				
	30%	20%	10%	30%	20%	10%	10%	10%
Foreground threshold	30%	20%	10%	30%	20%	10%	10%	10%
Background threshold	-	-	-	40%	40%	40%	50%	30%
0.0	0.55	0.55	0.55	0.55	0.55	0.55	0.55	0.55
0.1	0.33	0.33	0.28	0.3	0.3	0.27	0.26	0.28
0.2	0.17	0.18	0.15	0.13	0.15	0.15	0.15	0.14
0.3	0.09	0.1	0.09	0.07	0.08	0.1	0.09	0.08
0.4	0.03	0.04	0.02	0.03	0.05	0.04	0.03	0.03
0.5	0.01	0.01	0.0	0.01	0.02	0.02	0.02	0.0
0.6	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
0.7	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
0.8	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
0.9	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0

Table 15: The CAM to semantic mask generation methods results for multiple thresholds. While the Thresholded CAM mask generation produced on average better results at lower IOU scores, GrabCut produced better results at higher IOUs. Overall performance is still very low and any results with an IOU higher than 0.1 are too low to be useful since, we are looking for IOU score around 0.5.

representing semantic segmentation as first no individual objects have to be found as is the case with bounding boxes. The semantic segmentation masks are only being punished with IOU scores when missing an object, while in the case of bounding boxes this is regarded as an FN. We would, therefore, suspect that the bounding box methods can get better scores at higher IOU thresholds and this is also what we see. Comparing the thresholded CAMs with the Relative method we see almost no improvement.

5.4 CAM Loss Model Performance

In this section, we evaluate the performance of the CAM loss models. First, we evaluate the result on a image-wide label level. Second, we evaluate the models on the quality of the CAMs generated. Finally, we investigate the performance of bounding boxes and semantic segmentation masks generated on the CAMs.

We investigate if the CAM loss can solve the issue we had with the normal base method. We expect it to be able as the CAM loss is able to provide provide additional location information to the model. This information should also help CAMs find more objects as these masks are created out of each of the objects found within the view cone. This mask would then punish the CAMs for not finding more than 1 object. Finally, this should also disincentive the CAM fixing it on the most discriminative/selective features often also present within an image such as the case with traffic lights often being located near or on top of a particular stripped pole.

As the mixed range model showed the best overall performance, this model was used as our base model for the models with a CAM loss. For the CAM loss included models, we investigated using the last convolutional layer and using the last ‘add’ layer. Furthermore, we investigated two learning

Model	Convolutional	Add
Relative	0.56	0.59
Equal weights	0.48	0.47
Label less	0.4	0.59
Depth less	0.53	0.53

Table 16: Average F-score for the different CAM loss models at a learning rate 0.001. We see an overall performance increase when using the 'add' layer to generated CAMs.

rates for each of the different models, a learning rate of 0.01 and a learning rate of 0.001.

5.4.1 Classification

Table 16 illustrates the difference between using the 'add' layer versus using the last convolution layer (see appendix for comparison of all model switching between the 'add' layer and the convolutional layer 34).

We see an overall increase in performance when using the 'add' layer. This can be explained by the intuition that this 'add' layer will represent the overall model more than the last convolutional layer as the convolutional layer can be partially skipped by the 'add' layer. We, therefore, use this layer for all further experiments.

Next, we look at the classification result acquired by our CAM loss included models. Table 17 shows that the CAM loss model obtains overall worse or similar results to the normal Resnet model with a high learning rate. This indicates that our proposed CAM loss function does not improve image-wide classification performance.

Changing the normalization of the CAMs within the CAM loss changes the results slightly as illustrated in Table 18. Again, we do not see any significant improvement.

Model type	Non CAM loss	CAM loss							
Model name	Base model	Relative		Equal		Label less		Depth less	
Model learning rate	0.01	0.001	0.01	0.001	0.01	0.001	0.01	0.001	0.01
Average	0.66	-0.07	-0.05	-0.18	-0.03	-0.07	-0.08	-0.13	-0.05
Traffic light	0.74	0.0	-0.01	-0.04	-0.05	-0.04	-0.06	-0.04	-0.01
Barrier post	0.43	+0.18	0.0	+0.09	+0.17	+0.06	+0.13	+0.14	+0.1
Utility cabinet	0.62	-0.13	-0.05	-0.51	-0.04	-0.05	-0.09	-0.21	+0.02
Traffic sign	0.51	+0.08	+0.05	+0.06	-0.03	+0.05	-0.0	+0.06	+0.01
Manhole	0.52	-0.05	+0.11	-0.13	+0.06	-0.13	-0.03	-0.04	+0.03
Barrier/Gate	0.92	-0.02	+0.02	+0.05	-0.02	0.0	+0.02	0.0	+0.02
Street gully	0.28	-0.06	+0.05	-0.25	+0.06	-0.14	-0.21	-0.22	+0.08
Electronic freeway sign	0.89	-0.06	-0.06	0.0	0.0	+0.03	0.0	-0.03	-0.02
Bicycle stand	0.88	-0.1	-0.1	-0.1	-0.09	-0.09	-0.09	-0.09	-0.08
Street name sign	0.63	-0.05	-0.02	-0.28	+0.01	-0.06	-0.03	-0.19	-0.05
Garbage Can	0.52	-0.03	-0.07	-0.19	+0.01	-0.01	-0.04	-0.13	-0.03
Street light	0.75	+0.02	-0.01	-0.0	0.01	-0.0	-0.02	0.0	0.01
Camera	0.42	+0.1	+0.18	-0.02	+0.07	+0.14	+0.11	+0.04	+0.14
Bench	0.51	+0.08	-0.07	-0.06	+0.11	+0.02	+0.02	-0.06	+0.09
Roadside protection block	0.65	-0.05	+0.05	-0.57	+0.06	-0.07	-0.06	-0.4	-0.09
Information sign	0.3	+0.07	-0.07	-0.2	-0.02	+0.02	+0.03	-0.14	+0.01
Public play equipment	0.79	-0.03	+0.02	-0.11	-0.01	+0.02	-0.05	-0.03	0.01
Hectometer post	0.54	-0.35	-0.14	-0.24	-0.08	-0.22	-0.22	-0.24	-0.19
Ladder	0.57	-0.09	-0.03	-0.17	-0.03	0.0	-0.09	-0.12	-0.14
Roadsign direction Arrow	0.73	0.0	0.0	-0.06	+0.05	-0.01	+0.04	-0.01	+0.04

Table 17: F-scores of CAM loss models relative to baseline.

Model type	Non CAM loss												CAM loss																								
	Base model						Relative						Equal						Label less						Depth less												
	Normal	Confidence	Softmax	Normal	Confidence	Softmax	Normal	Confidence	Softmax	Normal	Confidence	Softmax	Normal	Confidence	Softmax	Normal	Confidence	Softmax	Normal	Confidence	Softmax	Normal	Confidence	Softmax													
Model name	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Normalization	0.66	0.74	0.43	0.51	0.52	0.92	0.28	0.89	0.88	0.63	0.52	0.75	0.42	0.51	0.65	0.3	0.79	0.54	0.57	0.73	0.66	0.62	0.55	0.73	0.42	0.51	0.65	0.3	0.79	0.54	0.57	0.73	0.66	0.62	0.55	0.73	
Average	-0.05	-0.08	-0.03	-0.05	-0.15	-0.04	0.0	-0.06	-0.04	-0.1	-0.25	-0.04	-0.01	-0.01	-0.01	-0.07	-0.11	-0.12	-0.03	-0.07	-0.03	-0.03	-0.07	-0.03	-0.01	-0.01	-0.01	-0.01	-0.01	-0.01	-0.01	-0.01	-0.01	-0.01	-0.01	-0.01	-0.01
Traffic light	0.0	+0.04	+0.14	+0.06	+0.04	+0.05	+0.07	+0.06	+0.06	0.0	0.0	0.0	+0.13	+0.17	+0.17	+0.06	+0.04	+0.05	+0.02	+0.02	+0.02	+0.02	+0.02	+0.02	+0.13	+0.17	+0.17	+0.06	+0.04	+0.05	+0.02	+0.02	+0.02	+0.02	+0.02	+0.02	
Barrier post	+0.05	+0.04	+0.02	+0.03	+0.04	+0.02	-0.12	-0.06	-0.04	-0.1	-0.09	-0.04	-0.03	-0.03	-0.03	-0.03	-0.03	-0.03	-0.03	-0.03	-0.03	-0.03	-0.03	-0.03	-0.03	-0.03	-0.03	-0.03	-0.03	-0.03	-0.03	-0.03	-0.03	-0.03	-0.03	-0.03	-0.03
Utility cabinet	+0.11	+0.03	-0.12	+0.06	+0.03	+0.05	+0.07	+0.06	+0.06	0.0	0.0	0.0	+0.13	+0.17	+0.17	+0.06	+0.04	+0.05	+0.02	+0.02	+0.02	+0.02	+0.02	+0.02	+0.13	+0.17	+0.17	+0.06	+0.04	+0.05	+0.02	+0.02	+0.02	+0.02	+0.02	+0.02	
Traffic sign	+0.05	+0.04	+0.02	+0.03	+0.04	+0.02	-0.12	-0.06	-0.04	-0.1	-0.09	-0.04	-0.03	-0.03	-0.03	-0.03	-0.03	-0.03	-0.03	-0.03	-0.03	-0.03	-0.03	-0.03	-0.03	-0.03	-0.03	-0.03	-0.03	-0.03	-0.03	-0.03	-0.03	-0.03	-0.03	-0.03	-0.03
Manhole	+0.02	+0.05	+0.05	+0.02	+0.02	+0.02	+0.02	+0.02	+0.02	0.0	0.0	0.0	+0.02	+0.02	+0.02	+0.02	+0.02	+0.02	+0.02	+0.02	+0.02	+0.02	+0.02	+0.02	+0.02	+0.02	+0.02	+0.02	+0.02	+0.02	+0.02	+0.02	+0.02	+0.02	+0.02	+0.02	+0.02
Barrier/Gate	+0.05	-0.06	+0.07	+0.06	-0.06	-0.04	0.0	0.0	0.0	0.0	0.0	0.0	-0.21	-0.28	-0.28	-0.21	-0.28	-0.28	-0.28	-0.28	-0.28	-0.28	-0.28	-0.28	-0.28	-0.28	-0.28	-0.28	-0.28	-0.28	-0.28	-0.28	-0.28	-0.28	-0.28	-0.28	-0.28
Street gully	-0.06	-0.04	0.0	0.0	-0.04	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
Electronic freeway sign	-0.1	-0.25	-0.1	-0.09	-0.09	-0.07	-0.09	-0.09	-0.09	-0.07	-0.09	-0.07	-0.09	-0.09	-0.09	-0.07	-0.09	-0.07	-0.09	-0.09	-0.09	-0.07	-0.09	-0.07	-0.09	-0.09	-0.09	-0.07	-0.09	-0.07	-0.09	-0.09	-0.09	-0.09	-0.07	-0.09	-0.07
Bicycle stand	-0.02	-0.04	+0.01	+0.01	-0.04	+0.01	+0.01	+0.01	+0.01	+0.01	+0.01	+0.01	+0.01	+0.01	+0.01	+0.01	+0.01	+0.01	+0.01	+0.01	+0.01	+0.01	+0.01	+0.01	+0.01	+0.01	+0.01	+0.01	+0.01	+0.01	+0.01	+0.01	+0.01	+0.01	+0.01	+0.01	+0.01
Street name sign	-0.07	-0.11	-0.03	+0.01	-0.11	-0.03	-0.03	-0.03	-0.03	-0.11	-0.03	-0.03	-0.11	-0.03	-0.11	-0.03	-0.03	-0.03	-0.11	-0.03	-0.11	-0.03	-0.03	-0.03	-0.11	-0.03	-0.11	-0.03	-0.03	-0.03	-0.03	-0.11	-0.03	-0.11	-0.03	-0.03	-0.03
Garbage Can	-0.01	-0.01	-0.01	0.01	-0.01	-0.01	-0.01	0.01	0.01	-0.01	-0.01	-0.01	-0.01	-0.01	-0.01	-0.01	-0.01	-0.01	-0.01	-0.01	-0.01	-0.01	-0.01	-0.01	-0.01	-0.01	-0.01	-0.01	-0.01	-0.01	-0.01	-0.01	-0.01	-0.01	-0.01	-0.01	-0.01
Street light	+0.18	+0.13	+0.08	+0.07	+0.14	+0.08	+0.08	+0.07	+0.14	+0.2	+0.14	+0.2	+0.14	+0.14	+0.14	+0.14	+0.14	+0.14	+0.14	+0.14	+0.14	+0.14	+0.14	+0.14	+0.14	+0.14	+0.14	+0.14	+0.14	+0.14	+0.14	+0.14	+0.14	+0.14	+0.14	+0.14	+0.14
Camera	-0.07	-0.03	+0.12	+0.11	-0.03	+0.12	+0.12	+0.11	+0.11	+0.19	+0.19	+0.19	+0.19	+0.19	+0.19	+0.19	+0.19	+0.19	+0.19	+0.19	+0.19	+0.19	+0.19	+0.19	+0.19	+0.19	+0.19	+0.19	+0.19	+0.19	+0.19	+0.19	+0.19	+0.19	+0.19	+0.19	+0.19
Bench	+0.05	+0.11	+0.02	+0.06	+0.11	+0.02	+0.02	+0.06	+0.06	-0.06	-0.06	-0.06	-0.06	-0.06	-0.06	-0.06	-0.06	-0.06	-0.06	-0.06	-0.06	-0.06	-0.06	-0.06	-0.06	-0.06	-0.06	-0.06	-0.06	-0.06	-0.06	-0.06	-0.06	-0.06	-0.06	-0.06	-0.06
Roadside protection block	-0.07	-0.14	-0.04	-0.02	-0.14	-0.04	-0.04	-0.02	-0.02	-0.02	-0.02	-0.02	-0.02	-0.02	-0.02	-0.02	-0.02	-0.02	-0.02	-0.02	-0.02	-0.02	-0.02	-0.02	-0.02	-0.02	-0.02	-0.02	-0.02	-0.02	-0.02	-0.02	-0.02	-0.02	-0.02	-0.02	-0.02
Information sign	+0.02	-0.11	+0.04	-0.01	-0.11	+0.04	+0.04	-0.01	-0.01	+0.02	+0.02	+0.02	+0.02	+0.02	+0.02	+0.02	+0.02	+0.02	+0.02	+0.02	+0.02	+0.02	+0.02	+0.02	+0.02	+0.02	+0.02	+0.02	+0.02	+0.02	+0.02	+0.02	+0.02	+0.02	+0.02	+0.02	+0.02
Public play equipment	-0.14	-0.12	-0.02	-0.08	-0.12	-0.02	-0.02	-0.08	-0.08	-0.04	-0.04	-0.04	-0.04	-0.04	-0.04	-0.04	-0.04	-0.04	-0.04	-0.04	-0.04	-0.04	-0.04	-0.04	-0.04	-0.04	-0.04	-0.04	-0.04	-0.04	-0.04	-0.04	-0.04	-0.04	-0.04	-0.04	-0.04
Hectometer post	-0.03	+0.05	+0.03	-0.03	-0.03	+0.03	+0.03	-0.03	-0.03	-0.03	-0.03	-0.03	-0.03	-0.03	-0.03	-0.03	-0.03	-0.03	-0.03	-0.03	-0.03	-0.03	-0.03	-0.03	-0.03	-0.03	-0.03	-0.03	-0.03	-0.03	-0.03	-0.03	-0.03	-0.03	-0.03	-0.03	-0.03
Ladder	0.0	+0.06	+0.14	+0.05	-0.02	+0.06	+0.06	+0.05	+0.05	-0.02	-0.02	-0.02	-0.02	-0.02	-0.02	-0.02	-0.02	-0.02	-0.02	-0.02	-0.02	-0.02	-0.02	-0.02	-0.02	-0.02	-0.02	-0.02	-0.02	-0.02	-0.02	-0.02	-0.02	-0.02	-0.02	-0.02	-0.02
Roadsign direction Arrow	0.66	0.74	0.43	0.51	0.52	0.92	0.28	0.89	0.88	0.63	0.52	0.75	0.42	0.51	0.65	0.3	0.79	0.54	0.57	0.73	0.66	0.62	0.55	0.73	0.42	0.51	0.65	0.3	0.79	0.54	0.57	0.73	0.66	0.62	0.55	0.73	

Table 18: Classification performance in F-scores for CAM loss models with high learning rate using the 'add' layer for different CAM normalization techniques.

These results on the CAM loss models seem to suggest that the information added by the CAM loss does not significantly increase our performance on an image-wide label level. To see the more direct impact of the CAM loss, we will investigate the accuracy of the CAMs generated by these methods in the next section.

5.4.2 CAM Performance

Table 19 shows the performance of the different CAM loss functions at the two different learning rates. While Table 20 shows the performance of the different CAM normalization methods. We, again, see a higher learning rate produce the best results. However, none of the CAM loss models show any significant improvement over the original best performing normal loss model. We do see a small improvement by the relative cam loss on a selected few objects i.e. Electronic freeway sign, public play equipment, ladders. Looking into the CAMs of the objects as illustrated by Figure 49 and Figure 50, we see two areas where major improvements are made: Firstly, the CAMs of the electronic freeway sign are less selective for the relative loss; Secondly, with public-play equipment, we see the model is better at distinguishing FPs where the normal loss model will sometimes learn the environment it is positioned in. The CAM loss models seem to be less incentivized to focus on a particular part, while also reducing the effects of learning the environment. It is interesting that these improvements, while visible in the other CAM loss models, lead to worse performance. For this we suspect two main reasons. First, this could be due to the fact that false-positive labels also get a mask created for them. While the image-wide labels also have this noise, the effect is worsened for these masks. This is as masks are generated from individual labels while image-wide labels are the grouped product of all these individual labels. This results in the image-wide label being more accurate than the individual labels as one TP label for a class can unintentionally correct all the FPs label for that class in an image. For example, if an image contains 20 street lights according to the map, but in actuality, only one of these street lights is visible, then image-wide labeling the image as containing a street light is correct and the model would be able to learn the appearance of a street light. However when generating masks this image would generate 19 FPs masks and only one TP mask. Second, as the CAM resolution is low, the loss is acquired on a large section of the image. This could disrupt the ability the network has to find the right features that should contribute more as such large areas are needed. This second CAM problem could be solved by increasing the CAM resolution which we investigate in section 5.5.

Model type	Non CAM loss	CAM loss							
Model name	Base model	Relative		Equal		Label less		Depth less	
Model learning rate	0.01	0.001	0.01	0.001	0.01	0.001	0.01	0.001	0.01
Average	0.33	-0.06	+0.03	-0.14	-0.04	-0.12	-0.05	-0.1	-0.05
Traffic light	0.34	-0.02	-0.0	-0.04	-0.05	-0.09	0.0	-0.0	-0.09
Barrier post	0.23	+0.05	-0.02	+0.03	+0.14	0.0	+0.15	+0.0	+0.12
Utility cabinet	0.29	-0.06	+0.08	-0.21	-0.07	-0.14	-0.0	-0.22	-0.08
Traffic sign	0.39	-0.08	-0.08	-0.15	-0.15	-0.14	-0.16	-0.16	-0.13
Manhole	0.34	-0.08	+0.03	-0.34	-0.05	-0.28	-0.04	-0.08	-0.07
Barrier/Gate	0.67	-0.13	+0.05	-0.11	-0.08	-0.09	-0.06	-0.07	-0.13
Street gully	0.25	-0.25	+0.04	-0.25	-0.04	-0.25	-0.25	-0.25	-0.04
Electronic freeway sign	0.29	-0.01	+0.19	-0.02	-0.02	-0.06	-0.0	-0.09	+0.08
Bicycle stand	0.17	-0.06	+0.06	-0.06	-0.05	-0.07	-0.08	-0.06	-0.09
Street name sign	0.0	0.0	+0.07	0.0	0.0	0.0	0.0	0.0	0.0
Garbage Can	0.48	-0.04	+0.03	-0.08	0.0	-0.08	-0.04	-0.08	-0.09
Street light	0.49	-0.04	-0.06	+0.03	-0.03	-0.01	-0.06	-0.06	-0.04
Camera	0.15	+0.0	+0.01	-0.09	-0.04	-0.04	-0.04	+0.05	+0.03
Bench	0.53	-0.03	+0.04	-0.17	-0.04	-0.12	-0.01	-0.09	+0.05
Roadside protection block	0.39	-0.01	+0.1	-0.26	-0.04	-0.24	-0.11	-0.25	-0.06
Information sign	0.13	-0.07	-0.13	-0.13	-0.01	-0.07	-0.07	-0.07	-0.08
Public play equipment	0.18	-0.09	+0.18	-0.11	-0.06	-0.04	+0.06	+0.03	-0.04
Hectometer post	0.52	-0.19	-0.14	-0.42	-0.19	-0.35	-0.19	-0.34	-0.14
Ladder	0.42	-0.13	+0.17	-0.34	+0.04	-0.21	-0.07	-0.16	-0.22
Roadsign direction Arrow	0.37	-0.04	-0.08	-0.01	-0.09	-0.05	-0.06	-0.07	-0.05

Table 19: CAM intersection score of the CAM loss models relative to baseline.

Model type	Non CAM loss			CAM loss												
	Model name	Base model		Relative			Equal			Label less			Depth less			
		Normal	Confidence	Softmax	Normal	Confidence	Softmax	Normal	Confidence	Softmax	Normal	Confidence	Softmax	Normal	Confidence	Softmax
Normalization	-															
Average	0.33	+0.03	-0.06	-0.04	-0.01	-0.06	-0.05	-0.08	-0.2	-0.05	-0.08	-0.2	-0.05	-0.02	-0.06	
Traffic light	0.34	-0.0	-0.12	-0.05	+0.11	-0.06	0.0	-0.01	-0.13	-0.09	-0.03	-0.07	-0.09	-0.03	-0.07	
Barrier post	0.23	-0.02	+0.06	+0.14	+0.08	+0.04	+0.15	+0.12	+0.01	-0.01	-0.07	+0.01	+0.07	+0.07	+0.09	
Utility cabinet	0.29	+0.08	-0.0	-0.07	+0.01	+0.01	-0.0	-0.01	-0.26	-0.08	-0.03	-0.06	-0.03	-0.06	-0.06	
Traffic sign	0.39	-0.08	-0.08	-0.15	-0.12	-0.12	-0.16	-0.08	-0.32	-0.13	-0.09	-0.32	-0.13	-0.09	-0.09	
Manhole	0.34	+0.03	-0.07	-0.05	-0.04	-0.09	-0.04	-0.09	-0.22	-0.07	+0.02	-0.22	-0.07	+0.02	-0.06	
Barrier/Gate	0.67	+0.05	-0.14	-0.08	+0.05	-0.11	-0.06	-0.06	-0.32	-0.13	-0.06	-0.32	-0.13	-0.24	-0.13	
Street gully	0.25	+0.04	-0.17	-0.04	-0.25	-0.25	-0.25	-0.25	-0.22	-0.04	-0.06	-0.22	-0.04	-0.06	+0.06	
Electronic freeway sign	0.29	+0.19	-0.01	+0.05	+0.12	+0.17	-0.0	+0.09	-0.08	+0.01	+0.01	-0.08	+0.08	+0.01	+0.06	
Bicycle stand	0.17	+0.06	-0.17	-0.1	-0.05	0.0	0.0	-0.06	-0.12	-0.09	-0.06	-0.12	-0.09	-0.06	-0.08	
Street name sign	0.0	+0.07	0.0	0.0	0.0	0.0	0.0	+0.06	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
Garbage Can	0.48	+0.03	-0.09	-0.15	0.0	+0.07	-0.04	-0.37	-0.29	-0.09	0.0	-0.29	-0.09	0.0	-0.04	
Street light	0.49	-0.06	-0.07	-0.11	-0.03	-0.07	-0.06	-0.04	-0.18	-0.04	-0.0	-0.18	-0.04	-0.0	-0.04	
Camera	0.15	+0.01	-0.02	-0.1	-0.04	+0.07	-0.15	-0.03	-0.11	-0.04	-0.03	-0.11	+0.03	-0.08	-0.1	
Bench	0.53	+0.04	-0.02	-0.01	-0.04	+0.06	-0.01	0.0	-0.21	+0.05	-0.01	-0.21	+0.05	-0.01	-0.02	
Roadside protection block	0.39	+0.1	0.0	-0.08	-0.04	-0.12	-0.11	-0.13	-0.28	-0.06	-0.06	-0.28	-0.06	-0.06	-0.12	
Information sign	0.13	-0.13	-0.02	-0.03	-0.01	-0.06	-0.07	-0.07	-0.1	-0.08	-0.06	-0.1	-0.08	-0.06	-0.03	
Public play equipment	0.18	+0.18	-0.01	+0.01	+0.04	+0.01	+0.06	-0.03	-0.07	-0.04	+0.16	-0.07	-0.04	+0.16	-0.07	
Hectometer post	0.52	-0.14	-0.16	-0.09	-0.19	+0.09	-0.24	-0.19	-0.41	-0.14	-0.12	-0.41	-0.14	-0.12	-0.17	
Ladder	0.42	+0.17	-0.02	-0.05	+0.04	+0.08	-0.08	-0.05	-0.35	-0.07	+0.12	-0.35	-0.07	+0.12	-0.15	
Roadsign direction Arrow	0.37	-0.08	-0.07	-0.09	-0.11	-0.1	-0.06	-0.06	-0.31	-0.05	+0.03	-0.31	-0.05	+0.03	-0.08	

Table 20: Comparison between the different CAM loss models and the base model by showing the relative increases and decrease in F-score performance using the CAM intersection metric with regards to the base model. All models use the ‘add’ layer for CAM generation and are trained using a high learning rate.

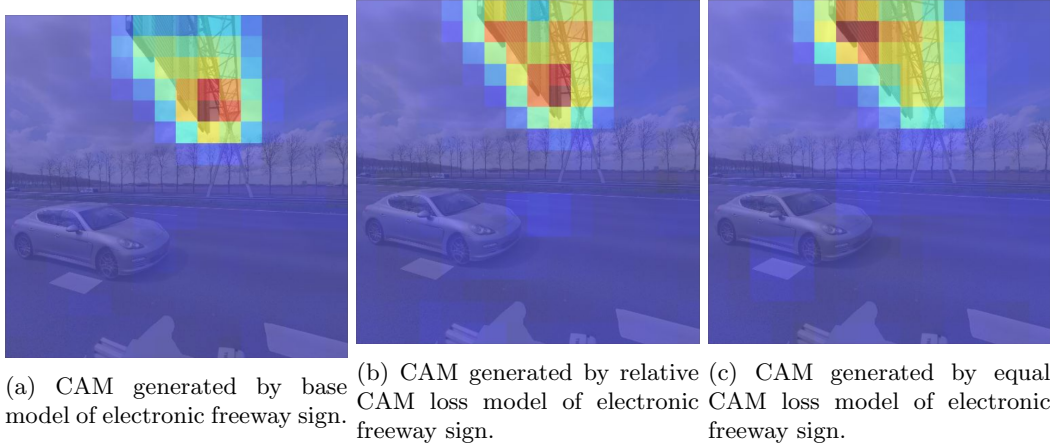


Figure 49: CAM of electronic freeway signs CAMs of the different models. While the equal loss has an overall worse performance, it does perform better in this example. In most cases, however, it over focuses on the entire images, taking into account the structure of overhang on which the electronic freeway signs are located.

5.4.3 Bounding Box And Semantic Segmentation Performance

Using the threshold found for the base model, we investigated the performance of our CAM loss models on a bounding box and semantic segmentation level. Where we use the best performing models i.e. model using the ‘add’ layer for CAM generation and with a high learning rate.

Table 21 illustrates our results generating bounding boxes without region proposals. Here we see a slight improvement using the relative CAM loss model when using the simple bounding box generation method. The rest of the CAM loss models seeming to have an overall worse or similar result to the Base model. Table 22 illustrates the result acquired when using bounding box generation with region proposals, here we see our base model still performs best which is unexpected as our relative model seems to have a better overall CAM. However since the overall performance differences are relatively small. We cannot adequately determine if our relative CAM loss model is indeed much better.

Tables 23 and 24 show the result required when generating masks from our CAMs generated by our CAM loss networks. Here, we again see the relative and base model produce the best results with our relative CAM loss model performing again slightly better. Furthermore, using GrabCut improves performance in most cases more than using the simple mask-generation method.

While we do see a slight increase in performance using our relative CAM loss method, it is the only CAM loss model to do so. This seems to indicate that the mask can indeed increase performance in cases where the model has issue in finding the object resulting in it taking the CAM loss more and so the mask more into account. A negative side effect of this is that this loss is only represent when our label is wrongly classified. Incases where we do correctly detect the object the mask is ignore. This result in the method less able in distuiquishing two objects are our often present within the images. For example, when a correct detection is made for the class Utility cabinets, which

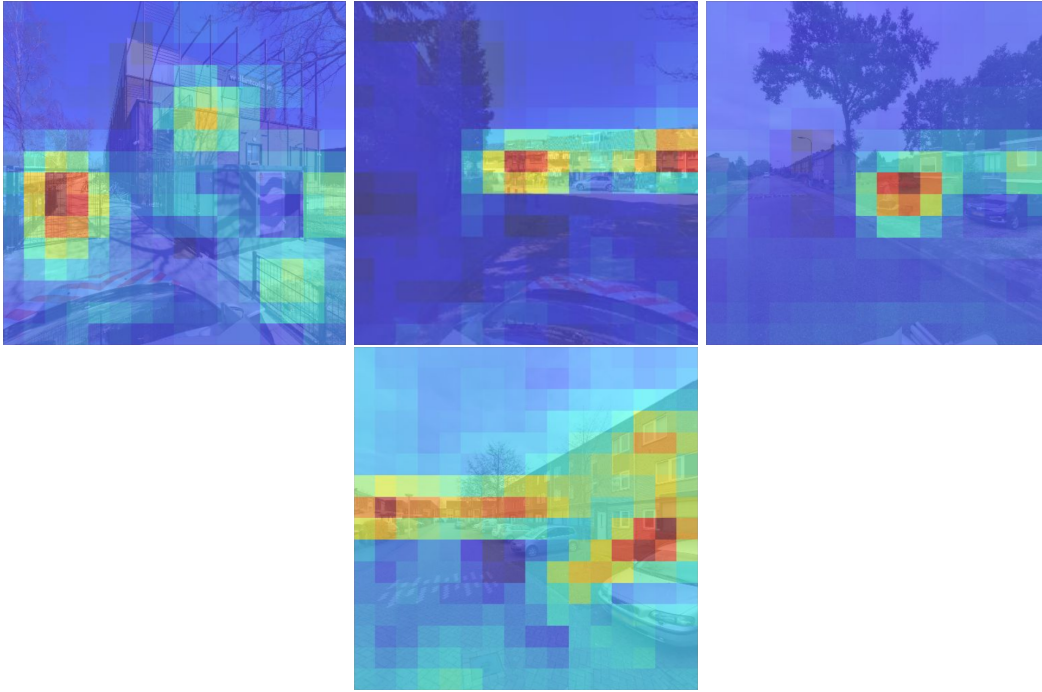


Figure 50: FP classification made by base model for public play equipment, mistakes not made by CAM loss models.

Add layer - learning rate 0.01							
Normalization	normal					Confidence	
CAM loss	Base model	Relative	Equal	Label less	Depth less	Equal	Depth less
0.0	0.55	0.55	0.54	0.54	0.54	0.54	0.55
0.1	0.25	0.28	0.22	0.24	0.25	0.23	0.25
0.2	0.16	0.18	0.13	0.14	0.15	0.13	0.16
0.3	0.1	0.13	0.09	0.09	0.09	0.09	0.09
0.4	0.06	0.1	0.05	0.05	0.05	0.04	0.04
0.5	0.03	0.04	0.02	0.01	0.01	0.02	0.03
0.6	0.01	0.02	0.01	0.0	0.0	0.0	0.01
0.7	0.0	0.0	0.01	0.0	0.0	0.0	0.0
0.8	0.0	0.0	0.0	0.0	0.0	0.0	0.0
0.9	0.0	0.0	0.0	0.0	0.0	0.0	0.0

Table 21: CAM loss models Fscores on different IOU thresholds for bounding box generation without region proposals. We see a slight increase in performance by the simple related loss function.

Normalization	Normal					Confidence	
CAM loss	Base model	Relative	Equal	Label less	Depth less	Equal	Depth less
0.0	0.55	0.55	0.55	0.54	0.54	0.53	0.55
0.1	0.17	0.14	0.15	0.14	0.16	0.13	0.15
0.2	0.11	0.07	0.09	0.06	0.1	0.06	0.08
0.3	0.09	0.05	0.07	0.06	0.06	0.05	0.06
0.4	0.08	0.04	0.04	0.05	0.04	0.04	0.05
0.5	0.05	0.03	0.03	0.04	0.03	0.03	0.04
0.6	0.03	0.02	0.01	0.02	0.02	0.03	0.02
0.7	0.02	0.01	0.01	0.02	0.02	0.02	0.01
0.8	0.01	0.01	0.0	0.01	0.0	0.01	0.01
0.9	0.0	0.0	0.0	0.0	0.0	0.0	0.0

Table 22: CAM loss models F-scores on different IOU thresholds for bounding box generation with region proposals. We see our base model observing the best performance.

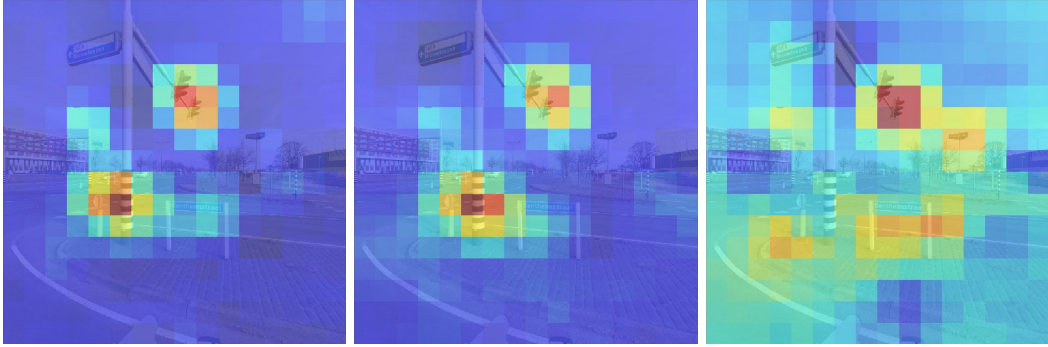
Add layer - learning rate 0.01							
Normalization	normal					Confidence	
CAM loss	Base model	Relative	Equal	Label less	Depth less	Equal	Depth less
0.0	0.53	0.52	0.53	0.52	0.53	0.51	0.52
0.1	0.33	0.37	0.32	0.32	0.34	0.29	0.31
0.2	0.18	0.21	0.14	0.15	0.17	0.15	0.16
0.3	0.1	0.12	0.08	0.06	0.07	0.07	0.07
0.4	0.04	0.04	0.04	0.04	0.03	0.02	0.03
0.5	0.01	0.01	0.01	0.0	0.0	0.0	0.0
0.6	0.0	0.0	0.0	0.0	0.0	0.0	0.0
0.7	0.0	0.0	0.0	0.0	0.0	0.0	0.0
0.8	0.0	0.0	0.0	0.0	0.0	0.0	0.0
0.9	0.0	0.0	0.0	0.0	0.0	0.0	0.0

Table 23: Semantic segmentation results, using a threshold that selects the top 20% of CAM to directly make a mask from the CAM.

add layer - learning rate 0.01							
Normalization	normal					Confidence	
CAM loss	Base model	Relative	Equal	Label less	Depth less	Equal	Depth less
0.0	0.53	0.52	0.53	0.52	0.53	0.51	0.52
0.1	0.31	0.34	0.31	0.32	0.32	0.28	0.29
0.2	0.19	0.23	0.17	0.18	0.18	0.16	0.17
0.3	0.11	0.14	0.12	0.1	0.12	0.1	0.09
0.4	0.07	0.07	0.07	0.05	0.07	0.05	0.06
0.5	0.04	0.04	0.05	0.01	0.02	0.03	0.03
0.6	0.03	0.02	0.02	0.0	0.02	0.01	0.01
0.7	0.01	0.01	0.01	0.0	0.0	0.0	0.01
0.8	0.01	0.0	0.0	0.0	0.0	0.0	0.0
0.9	0.0	0.0	0.0	0.0	0.0	0.0	0.0

Table 24: Semantic segmentation results using the Grabcut method between the Base model, relative loss model, equal loss model and depth less model.

are often located next to electronic freeway sign, the CAM loss is unable to divert the activation of the CAM from the electronic freeway sign to the actual location of the Utility Cabinet. This can be seen in Figure 51, this also shows that the equal model is able to distinguish the two more as it does take into account the CAM loss even when a positive classification is made. However, this equal approach also introduces downside as the mask has an increased amount of label noise within them, resulting in overall lower scores. Furthermore, as the base and relative model find overall more objects, their F-scores, when the IOU is low, are on average higher than the CAM loss functions. As such the scores at higher IOU's are equalized. Having said that, the CAM loss models are often still not able to find the object although they seem to get closer. The faults made by the CAMs also indicate an alternative issue with the automatically generated masks. Objects that are underneath or positioned within the same depth range and same direction range cannot be distinguished leading to CAM where the mask cannot adequately distinguish objects. As such the masks do help to distinguish traffic signs from traffic lights and change the CAM of utility closets from focusing on matrix signs which is a good thing however they instead also focus on the overhang, which is also present within the depth masks.



(a) CAM generated by base model. (b) CAM generated by relative model. (c) CAM generated by equal model.

Figure 51: CAMs illustrating that the equal model is the only model capable of removing the discriminative detection on the black and white pole which is often also present within images where traffic light are also present.

5.5 Upscaled CAM loss models

In this section, we evaluate the performance of the upscaled CAM loss models. First, we evaluate the result on a image-wide label level. Second, we evaluate the models on the quality of the CAMs generated. Finally, we investigate the performance of bounding boxes and semantic-segmentation masks generated on the CAMs.

5.5.1 Classification

Using our best performing CAM loss model, more specifically the relative CAM loss and the equal loss models as it more generally uses the CAM information, we investigate upscaling the CAMs in an effort to increase the resolution and expressiveness of the CAMs. These upscaled models are trained using a high learning rate and generate CAMs using the ‘add’. We upscale these models to two different increased sized. First to a CAM size of 32×32 and second to a CAM size of 64×64 . We thereby increase the size of the CAM from 16×16 images to $32 \times 32/64 \times 64$ images.

Table 25 illustrates our classification result acquired using our upscaled models. While the 32×32 model relative model shows a small increase in performance over itself the rest of the upscaled models show a significant drop in performance by increasing the size of the CAMs. This can be described to the fact that the networks now have to learn an increased number of parameters taking longer to train which leads to the noise in the datasets increasing its effect. This is also the reason, we see large decrease in the more noisy classes such as information signs. While this is less the case for more stable objects such as Barrier/gate.

Loss	normal	relative			equal		
		16 × 16	16 × 16	32 × 32	64 × 64	16 × 16	32 × 32
CAM size	16 × 16	16 × 16	32 × 32	64 × 64	16 × 16	32 × 32	64 × 64
Average	0.66	0.6	0.63	0.57	0.62	0.55	0.55
Traffic light	0.74	0.72	0.77	0.61	0.69	0.72	0.59
Barrier post	0.43	0.43	0.57	0.57	0.6	0.54	0.55
Utility cabinet	0.62	0.57	0.59	0.34	0.58	0.38	0.58
Traffic sign	0.51	0.56	0.45	0.53	0.49	0.51	0.5
Manhole	0.52	0.63	0.65	0.54	0.58	0.36	0.45
Barrier/Gate	0.92	0.95	0.87	0.94	0.9	0.9	0.9
Street gully	0.28	0.33	0.45	0.13	0.34	0.13	0.21
Electronic freeway sign	0.89	0.83	0.84	0.84	0.89	0.89	0.91
Bicycle stand	0.88	0.78	0.78	0.81	0.79	0.73	0.75
Street name sign	0.63	0.61	0.54	0.63	0.64	0.49	0.37
Garbage Can	0.52	0.44	0.52	0.29	0.53	0.5	0.47
Street light	0.75	0.74	0.75	0.74	0.76	0.78	0.75
Camera	0.42	0.59	0.55	0.5	0.48	0.35	0.47
Bench	0.51	0.44	0.56	0.38	0.62	0.67	0.55
Roadside protection block	0.65	0.7	0.7	0.8	0.71	0.65	0.64
Information sign	0.3	0.24	0.07	0.17	0.29	0.12	0.13
Public play equipment	0.79	0.82	0.81	0.67	0.78	0.7	0.7
Hectometer post	0.54	0.4	0.63	0.43	0.45	0.33	0.32
Ladder	0.57	0.54	0.67	0.63	0.55	0.52	0.5
Roadsign direction Arrow	0.73	0.73	0.79	0.87	0.78	0.67	0.63

Table 25: Classification results in fscores using upscaled CAM models, we see a small improvement when upscaling the relative model to a size of 32×32 . This seems to be mostly due to some of the smaller objects now receiving more accurate CAM losses as the size of the 16×16 would often generate high losses even when fixated on the right location as the resolution was off.

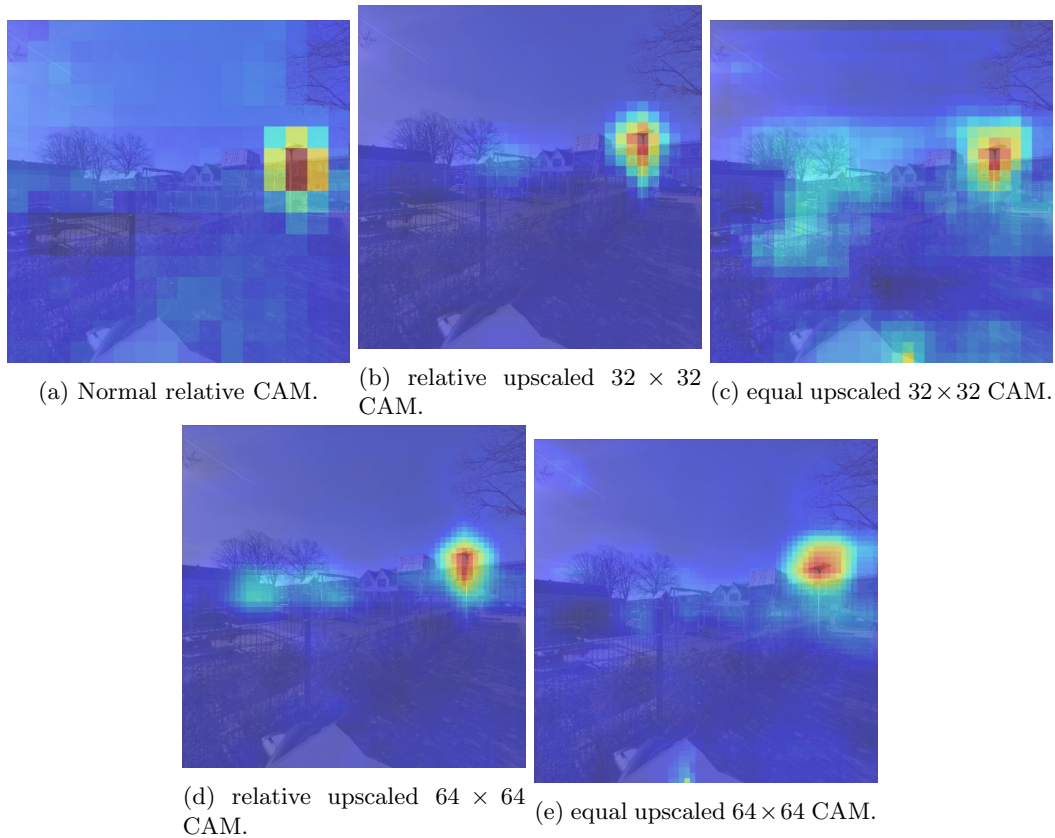


Figure 52: CAMs generated by upscaled models for the detection of a street light.

5.5.2 CAM Performance

Our upscaled models CAMs should allow for increased resolution and expressiveness, figures 52 and 53 illustrate the result acquired by the upscaling of the CAMs. Where we can see that the increase of resolution does allow the model to focus on more particular features. We also see the relative model being able to find the Utility cabinet and distinguish it from an electronic freeway sign. One can also see that the equal model has been affected more by the depth masks as it still responds heavily to the structure above the utility cabinets. This region is almost always also included when making the depth mask for this object as it is situated in the same direction and at the same distance.

Looking now at how many of the CAM intersect with the ground truth, surprisingly we only see a slight decrease in performance as shown in Table 26. This is surprising as the CAM generated by the upscaled models are on average smaller and more focused which decreases the chance of them overlapping with the ground truth.

Loss	normal	relative			equal		
		16×16	32×32	64×64	16×16	32×32	64×64
CAM size	16×16	16×16	32×32	64×64	16×16	32×32	64×64
Average	0.42	0.38	0.36	0.34	0.34	0.31	0.31
Traffic light	0.43	0.41	0.37	0.45	0.32	0.44	0.43
Barrier post	0.32	0.42	0.38	0.42	0.39	0.45	0.17
Utility cabinet	0.3	0.25	0.33	0.21	0.27	0.35	0.2
Traffic sign	0.41	0.39	0.25	0.35	0.25	0.3	0.34
Manhole	0.41	0.3	0.4	0.38	0.33	0.31	0.34
Barrier/Gate	0.83	0.83	0.83	0.89	0.7	0.82	0.77
Street gully	0.26	0.03	0.2	0.0	0.23	0.07	0.13
Electronic freeway sign	0.52	0.45	0.51	0.36	0.3	0.42	0.55
Bicycle stand	0.25	0.25	0.0	0.29	0.18	0.22	0.17
Street name sign	0.0	0.0	0.0	0.11	0.0	0.0	0.07
Garbage Can	0.61	0.61	0.6	0.35	0.53	0.48	0.42
Street light	0.5	0.47	0.47	0.39	0.47	0.29	0.4
Camera	0.24	0.18	0.11	0.23	0.14	0.08	0.18
Bench	0.62	0.61	0.5	0.36	0.54	0.34	0.51
Roadside protection block	0.54	0.65	0.49	0.44	0.4	0.39	0.32
Information sign	0.15	0.08	0.08	0.09	0.15	0.09	0.06
Public play equipment	0.2	0.15	0.41	0.38	0.18	0.26	0.22
Hectometer post	0.7	0.48	0.48	0.35	0.44	0.22	0.31
Ladder	0.48	0.5	0.38	0.27	0.63	0.14	0.35
Roadsign direction Arrow	0.55	0.51	0.44	0.45	0.33	0.51	0.33

Table 26: CAM intersection performance for upscaled models. Where a prediction is seen as positive if the top 20% of the CAM intersect with the Semantic segmentation groundtruth. Here we see an overall decrease in performance by upscaling the models. While initially unexpected, it does follow the intuition that as we increase the resolution the CAM has to be more precise to still intersect with the ground truth. For examples, at the 16×16 resolution a pixel can easily overlap with the ground truth while at a resolution of 32×32 the models has to be more precise to still intersect. This can explain why our result show an decrease in performance. Furthermore, as we only take the top 20% by increasing the number of pixels we decrease the size of the activated region.

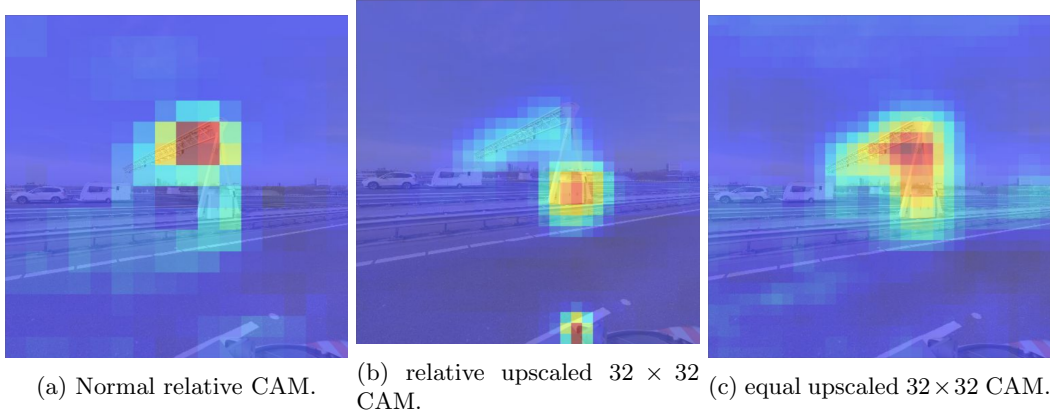


Figure 53: CAMS generated by upscaled models for the detection of a utility cabinets. Note no position classification was made by the 64×64 models and as such are not shown.

5.5.3 Bounding Box And Semantic Segmentation Performance

Tables 27 and 28 show the bounding box results acquired by the upscaled models. Unfortunately, we do not see any major improvement with any of the upscaled models. This seems to indicate that we can upscale the CAMs and increase the resolution for certain objects. However, this does make the learning process harder and can, in fact, make the noise present within the depth generated masks more significant, as shown in Figure 53. This same result is also shown when generating semantic mask as shown by Tables 29 and 30.

6 Conclusions

We set out to investigate if the semantic information present with certain maps could be used to construct an image-wide labeled dataset with which one could train a SOTA object detection method. Furthermore, we set out to find what kind of result could be acquired on such a dataset i.e. what kind of bounding box and semantic segmentation result could be acquired. For this, we investigated using the weakly supervised technique of CAMs. To improve the accuracy of the CAMs, we investigated two methods the first was simply to change which layer was used for the CAM generation, which as we were using a Resnet network gave us a choice. Second, we also investigate if we could automatically generate object masks by combining depth information with semantic information available from the maps and use these masks to improve the CAMs.

What we found was a method that while capable of automatically generating large image-wide datasets, was full of noisy labels. This noise mostly consists of label issues, labels not being visible due to dynamic obstacles, map information not accurately describing what can be seen within an image due to changes within the time of measurement and the time of image capture, and overall, some noise within the maps themselves. Regardless of this high amount of noise, a Resnet 50 network was capable of producing an average F-score of 0.66. Which when taking into account

Loss	normal	relative			equal		
CAM size	16 × 16	16 × 16	32 × 32	64 × 64	16 × 16	32 × 32	64 × 64
0.0	0.55	0.55	0.54	0.51	0.54	0.47	0.49
0.1	0.25	0.28	0.26	0.22	0.23	0.19	0.19
0.2	0.16	0.18	0.15	0.13	0.13	0.11	0.11
0.3	0.1	0.13	0.11	0.08	0.09	0.06	0.06
0.4	0.06	0.1	0.07	0.05	0.04	0.03	0.03
0.5	0.03	0.04	0.04	0.01	0.02	0.02	0.01
0.6	0.01	0.02	0.02	0.0	0.0	0.01	0.0
0.7	0.0	0.0	0.0	0.0	0.0	0.0	0.0
0.8	0.0	0.0	0.0	0.0	0.0	0.0	0.0
0.9	0.0	0.0	0.0	0.0	0.0	0.0	0.0

Table 27: Resulting Fscores for bounding box generation, without region proposals on different IOU thresholds. Here we see the upscaled models perform worse then we would have expected. We suspect that this is due to two reasons. First, by increasing the resolution we need to learn more parameters. Second, due to the increased resolution the generated CAMs will be more able to focus on the most discriminative/selective features, resulting in it ignoring other key areas of the objects necessary for generating accurate bounding boxes.

Loss	normal	relative			equal		
CAM size	16 × 16	16 × 16	32 × 32	64 × 64	16 × 16	32 × 32	64 × 64
0.0	0.55	0.55	0.54	0.52	0.53	0.47	0.49
0.1	0.16	0.14	0.14	0.12	0.13	0.1	0.11
0.2	0.11	0.07	0.08	0.07	0.06	0.07	0.07
0.3	0.06	0.05	0.06	0.06	0.05	0.05	0.05
0.4	0.05	0.04	0.05	0.03	0.04	0.03	0.04
0.5	0.05	0.03	0.03	0.02	0.03	0.03	0.03
0.6	0.04	0.02	0.02	0.02	0.03	0.01	0.02
0.7	0.03	0.01	0.01	0.01	0.02	0.01	0.01
0.8	0.01	0.01	0.0	0.0	0.01	0.0	0.01
0.9	0.0	0.0	0.0	0.0	0.0	0.0	0.0

Table 28: Resulting Fscores for bounding box generation, with region proposals on different IOU thresholds. Here we see similar results as bounding box generation without region proposals. Only shifted towards receiving slightly better scores at higher IOU’s at the cost of lower scores at the lower IOU ranges.

Loss	normal	relative			equal		
CAM size	16 × 16	16 × 16	32 × 32	64 × 64	16 × 16	32 × 32	64 × 64
0.0	0.53	0.52	0.52	0.5	0.53	0.47	0.47
0.1	0.33	0.37	0.33	0.29	0.32	0.1	0.28
0.2	0.18	0.21	0.21	0.17	0.14	0.07	0.15
0.3	0.1	0.12	0.12	0.09	0.08	0.05	0.08
0.4	0.04	0.04	0.08	0.05	0.04	0.03	0.01
0.5	0.01	0.01	0.04	0.0	0.01	0.03	0.0
0.6	0.0	0.0	0.0	0.0	0.0	0.01	0.0
0.7	0.0	0.0	0.0	0.0	0.0	0.01	0.0
0.8	0.0	0.0	0.0	0.0	0.0	0.0	0.0
0.9	0.0	0.0	0.0	0.0	0.0	0.0	0.0

Table 29: Results Fscores semantic mask generation, without GrabCut at different IOU thresholds. Interestingly, we see the best results at a the higher IOU threshold yet. Both the relative and equal 32 × 32 upscaled models able to outperform at a IOU threshold 0.5. Intuiatlifly this is due the model able to beter respresent some of the objects with its higher resolution specficly smaller objects that due not suffer as much from the CAM having the ability to focus on the most discriminative/selective features.

Loss	normal	relative			equal		
CAM size	16 × 16	16 × 16	32 × 32	64 × 64	16 × 16	32 × 32	64 × 64
0.0	0.53	0.52	0.52	0.5	0.51	0.45	0.47
0.1	0.31	0.34	0.31	0.27	0.28	0.25	0.26
0.2	0.19	0.23	0.2	0.16	0.16	0.15	0.18
0.3	0.11	0.14	0.13	0.11	0.1	0.09	0.12
0.4	0.07	0.07	0.1	0.07	0.05	0.04	0.07
0.5	0.04	0.04	0.05	0.03	0.03	0.03	0.03
0.6	0.03	0.02	0.03	0.02	0.01	0.02	0.02
0.7	0.01	0.01	0.03	0.01	0.0	0.0	0.01
0.8	0.01	0.0	0.0	0.0	0.0	0.0	0.0
0.9	0.0	0.0	0.0	0.0	0.0	0.0	0.0

Table 30: Results Fscores semantic mask generation, with GrabCut at different IOU thresholds. Here we see similar result to the mask generation with Grabcut. The relative 16 × 16 model able to out perform the rest of the models at the higher IOU scales. However, unlike before the equal loss models doesn't perform as well as before. We expect this to be due to the grabcut method expending the foreground into not background regions. These are expected to especially more noisy in the equal model as these are probably the regions also falsely covered by the approximate masks. Resulting in a decrease in performance.

that the average F-score of the dataset the network trained on was only 0.64 is a very good result. Specify certain classes we very successfully i.e. 'barrier/gate', 'bicycle stands', 'electronic freeway signs'. While the network had more performance problems with smaller object i.e. 'street gully', 'hectometer post'. These are common problems with object detection and numerous methods exist to improve the ability of a network for detecting smaller objects (76). These objects could increase the performance of the method was more specialized for smaller objects.

While these results showed promise, converting them to accurate bounding boxes and semantic segmentation masks results proved to be a challenging task. To generate these mask the CAM method was used. However, while the normal CAM methods state taking the last convolutional layer of a network to generate CAMs with, a Resnet network as a layer after the last convolutional layer which should be able to represent the features that led to a classification better. This other layer the 'add' layer was indeed able to give slightly better CAMs. However even with this slight increase in performance numerous faults were still present within the CAMs. We quickly discovered that certain objects would pair in CAMs with other objects also present within these environments. For example, Utility cabinets and electronic freeway signs often located within the same environment would, in a decent number of cases, generate identical CAMs. Where the utility cabinet would match the electronic freeway sign as it is easier to discriminate and find the utility cabinet is. Furthermore, it became clear that the network would almost always only focus on the most discriminative parts of an object of the most discriminative object in the image even if multiple objects were present within an image. This resulted in overall poor performance when generating bounding boxes and semantic masks.

To improve these results, we introduced a CAM loss that uses the automatically generate mask to guide the network into localizing the correct features and correct objects. To incorporate this CAM loss within the Resnet 50 network we investigated multiple combined losses where we found the relative loss, which takes the CAM loss more into account when the normal loss makes a bigger mistake, to have the best performance. While the overall performance of the CAM loss model was worse or equal to the normal loss model. The relative model was able to get a slight increase in performance, where the average F-score of our cam intersection metric showed an increase of 0.03 and an increase of 7% could be seen at a semantic segmentation level. While small it shows that our method is capable of improving the CAMs. However, in most cases the method introduces more problems than it would solve. For which we found two main reasons: first as the mask method uses all labels for each image to generate mask instead of grouping them together to a single label per class. It introduces more of the noise that is within the dataset. Second, the masks themselves depend on the object being alone by itself as for instance when placed on a wall or other large surface our method of generating masks fails. These two issues introduced numerous new errors within the learning process reducing overall performance. Finally, we also investigated upscaling the CAM models using the automatically generated mask. Here, at the higher IOU threshold ranges we found a small increase in performance the increased resolution able to better represent some of the objects. However, in most cases, they perform worse due the upscaling relying more heavily on the mask which as outlined above showed numerous faults.

In conclusion, we found that we were able to automatically generate object detection datasets, on which the SOTA object detection method could learn and acquire decent image-wide classification results. However, results for the bounding box and semantic segmentation were insufficient. While small improvements could be seen by leveraging depth information with semantic map information, the increase was too small to allow for usable bounding box and semantic segmentation results. The

method does, however, show promise, and with further research could lead to more usable results.

7 Future Work

As the main issue with our method is dealing with a large amount of label noise within the dataset. We suggest a semi-supervised method could help solve some of the major issues present within using a map as a form of automatic labeling.

Furthermore, the noisy generated mask could be improved by using multiple viewpoints to construct a more accurate mask of the object when possible. For instance, from a particular point of view, an object might disappear within the background of another object especially when using depth information. Using multiple viewpoints the possibility exists to further refine and improve these depth masks.

In our approach to place a loss on the CAM, we automatically generated a mask in order to help focus the CAMs. However, one could also step away from the weakly supervised paradigm and use these generate masks directly within a common bounding box method. For this to have any chance of working the map labels or the mask then self should be filtered in some way to reduce the noise within them. This could be done manually but other techniques such as clustering the generated masks could prove to be successful in removing large amounts of noisy.

As stated in our research the CAMs focus too much on the most discriminative features within an image. A possible solution would be to filter the most activated regions out and try to let the network learn on the remaining features. This would make the CAMs less discriminative and allow them to generate more encompassing CAMs. This approach as suggested by Wei et al. (52) and shows promising results that could help solve some of the CAM issues.

As we also saw problems with too few negative examples within certain environments. This could probably be solved or the severity of this problem can be reduced by changing the method to specifically look for images that do not contain these objects within an environment. This could reduce the focus on background images without needing manual supervision.

Finally, in our approach, we did not consider objects being blocked by other objects that are also labeled on maps. For instance, there are maps that show the shape of buildings, this information could be used to better filter which labels are visible. For example, if a street gully is behind a row of houses it will probably not be visible to the camera.

8 Bib

References

- [1] 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).
- [2] Liu, L., Ouyang, W., Wang, X., Fieguth, P., Chen, J., Liu, X., & Pietikäinen, M. (2018). Deep learning for generic object detection: A survey. *arXiv preprint arXiv:1809.02165*.

- [3] Dai, J., He, K., & Sun, J. (2015). Boxesup: Exploiting bounding boxes to supervise convolutional networks for semantic segmentation. In Proceedings of the IEEE International Conference on Computer Vision (pp. 1635-1643).
- [4] Crowley, E. J., & Zisserman, A. (2016, October). The art of detection. In European Conference on Computer Vision (pp. 721-737). Springer, Cham.
- [5] Zadrija, V., Krapac, J., Šegvić, S., & Verbeek, J. (2018). Sparse weakly supervised models for object localization in road environment. *Computer Vision and Image Understanding*.
- [6] Girshick, R., Donahue, J., Darrell, T., & Malik, J. (2014). Rich feature hierarchies for accurate object detection and semantic segmentation. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 580-587).
- [7] Uijlings, J. R., Van De Sande, K. E., Gevers, T., & Smeulders, A. W. (2013). Selective search for object recognition. *International journal of computer vision*, 104(2), 154-171.
- [8] He, K., Zhang, X., Ren, S., & Sun, J. (2014, September). Spatial pyramid pooling in deep convolutional networks for visual recognition. In European conference on computer vision (pp. 346-361). Springer, Cham.
- [9] Girshick, R. (2015). Fast r-cnn. In Proceedings of the IEEE international conference on computer vision (pp. 1440-1448).
- [10] Wu, J., Yu, Y., Huang, C., & Yu, K. (2015). Deep multiple instance learning for image classification and auto-annotation. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (pp. 3460-3469).
- [11] Dai, J., Li, Y., He, K., & Sun, J. (2016). R-fcn: Object detection via region-based fully convolutional networks. In Advances in neural information processing systems (pp. 379-387).
- [12] He, K., Gkioxari, G., Dollár, P., & Girshick, R. (2017, October). Mask r-cnn. In *Computer Vision (ICCV), 2017 IEEE International Conference on* (pp. 2980-2988). IEEE.
- [13] Zhou, Y., Zhu, Y., Ye, Q., Qiu, Q., & Jiao, J. (2018). Weakly Supervised Instance Segmentation using Class Peak Response. *arXiv preprint arXiv:1804.00880*.
- [14] Chen, L. C., Papandreou, G., Kokkinos, I., Murphy, K., & Yuille, A. L. (2018). Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. *IEEE transactions on pattern analysis and machine intelligence*, 40(4), 834-848.
- [15] 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).
- [16] Simonyan, K., & Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*.
- [17] Schmidhuber, J. (2015). Deep learning in neural networks: An overview. *Neural networks*, 61, 85-117.
- [18] Szegedy, C., Toshev, A., & Erhan, D. (2013). Deep neural networks for object detection. In Advances in neural information processing systems (pp. 2553-2561).

- [19] Wei, Y., Shen, Z., Cheng, B., Shi, H., Xiong, J., Feng, J., & Huang, T. (2018, July). TS2C: tight box mining with surrounding segmentation context for weakly supervised object detection. In *European Conference on Computer Vision* (pp. 454-470). Springer, Cham.
- [20] Bilen, H., & Vedaldi, A. (2016). Weakly supervised deep detection networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (pp. 2846-2854).
- [21] Cinbis, R. G., Verbeek, J., & Schmid, C. (2017). Weakly supervised object localization with multi-fold multiple instance learning. *IEEE transactions on pattern analysis and machine intelligence*, 39(1), 189-203.
- [22] Jie, Z., Wei, Y., Jin, X., Feng, J., & Liu, W. (2017, July). Deep self-taught learning for weakly supervised object localization. In *IEEE CVPR* (Vol. 2).
- [23] Lai, B., & Gong, X. (2017). Saliency guided end-to-end learning for weakly supervised object detection. *arXiv preprint arXiv:1706.06768*.
- [24] Liang, X., Liu, S., Wei, Y., Liu, L., Lin, L., & Yan, S. (2015). Towards computational baby learning: A weakly-supervised approach for object detection. In *Proceedings of the IEEE international conference on computer vision* (pp. 999-1007).
- [25] Hastie, T., Tibshirani, R., & Friedman, J. (2009). Unsupervised learning. In *The elements of statistical learning* (pp. 485-585). Springer, New York, NY.
- [26] Wang, X., & Gupta, A. (2015). Unsupervised learning of visual representations using videos. In *Proceedings of the IEEE International Conference on Computer Vision* (pp. 2794-2802).
- [27] Socher, R., Ganjoo, M., Manning, C. D., & Ng, A. (2013). Zero-shot learning through cross-modal transfer. In *Advances in neural information processing systems* (pp. 935-943).
- [28] Lampert, C. H., Nickisch, H., & Harmeling, S. (2014). Attribute-based classification for zero-shot visual object categorization. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 36(3), 453-465.
- [29] Erhan, D., Bengio, Y., Courville, A., Manzagol, P. A., Vincent, P., & Bengio, S. (2010). Why does unsupervised pre-training help deep learning?. *Journal of Machine Learning Research*, 11(Feb), 625-660.
- [30] Tang, P., Wang, X., Bai, X., & Liu, W. (2017). Multiple instance detection network with online instance classifier refinement. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (pp. 2843-2851).
- [31] Bilen, H., Pedersoli, M., & Tuytelaars, T. (2015). Weakly supervised object detection with convex clustering. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (pp. 1081-1089).
- [32] Song, H. O., Girshick, R., Jegelka, S., Mairal, J., Harchaoui, Z., & Darrell, T. (2014). On learning to localize objects with minimal supervision. *arXiv preprint arXiv:1403.1024*.
- [33] Gokberk Cinbis, R., Verbeek, J., & Schmid, C. (2014). Multi-fold mil training for weakly supervised object localization. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 2409-2416).

- [34] Zhou, B., Khosla, A., Lapedriza, A., Oliva, A., & Torralba, A. (2016). Learning deep features for discriminative localization. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (pp. 2921-2929).
- [35] Ahn, J., & Kwak, S. (2018). Learning Pixel-level Semantic Affinity with Image-level Supervision for Weakly Supervised Semantic Segmentation. arXiv preprint arXiv:1803.10464.
- [36] Fan, J., Zhang, Z., & Tan, T. (2018). CIAN: Cross-Image Affinity Net for Weakly Supervised Semantic Segmentation. arXiv preprint arXiv:1811.10842.
- [37] Zhang, D., Meng, D., Zhao, L., & Han, J. (2017). Bridging saliency detection to weakly supervised object detection based on self-paced curriculum learning. arXiv preprint arXiv:1703.01290.
- [38] Zhu, W., Liang, S., Wei, Y., & Sun, J. (2014). Saliency optimization from robust background detection. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 2814-2821).
- [39] Andrews, S., Tsochantaridis, I., & Hofmann, T. (2003). Support vector machines for multiple-instance learning. In Advances in neural information processing systems (pp. 577-584).
- [40] Wu, J., Yu, Y., Huang, C., & Yu, K. (2015). Deep multiple instance learning for image classification and auto-annotation. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (pp. 3460-3469).
- [41] Scherer, D., Müller, A., & Behnke, S. (2010). Evaluation of pooling operations in convolutional architectures for object recognition. In Artificial Neural Networks–ICANN 2010 (pp. 92-101). Springer, Berlin, Heidelberg.
- [42] Liu, Y., Gadepalli, K., Norouzi, M., Dahl, G. E., Kohlberger, T., Boyko, A., ... & Hipp, J. D. (2017). Detecting cancer metastases on gigapixel pathology images. arXiv preprint arXiv:1703.02442.
- [43] Lin, T. Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., ... & Zitnick, C. L. (2014, September). Microsoft coco: Common objects in context. In European conference on computer vision (pp. 740-755). Springer, Cham.
- [44] Deng, J., Dong, W., Socher, R., Li, L. J., Li, K., & Fei-Fei, L. (2009, June). Imagenet: A large-scale hierarchical image database. In Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on (pp. 248-255). Ieee.
- [45] Anthony, M. (2001). Discrete mathematics of neural networks: selected topics (Vol. 8). Siam.
- [46] Cortes, C., & Vapnik, V. (1995). Support-vector networks. Machine learning, 20(3), 273-297.
- [47] Itti, L., Koch, C., & Niebur, E. (1998). A model of saliency-based visual attention for rapid scene analysis. IEEE Transactions on pattern analysis and machine intelligence, 20(11), 1254-1259.
- [48] Deselaers, T., Alexe, B., & Ferrari, V. (2012). Weakly supervised localization and learning with generic knowledge. International journal of computer vision, 100(3), 275-293.

- [49] Nguyen, M. H., Torresani, L., De La Torre, F., & Rother, C. (2009, September). Weakly supervised discriminative localization and classification: a joint learning process. In *Computer Vision, 2009 IEEE 12th International Conference on* (pp. 1925-1932). IEEE.
- [50] Bilen, H., Pedersoli, M., & Tuytelaars, T. (2014, January). Weakly supervised object detection with posterior regularization. In *Proceedings BMVC 2014* (pp. 1-12).
- [51] Sangineto, E., Nabi, M., Culibrk, D., & Sebe, N. (2019). Self paced deep learning for weakly supervised object detection. *IEEE transactions on pattern analysis and machine intelligence*, 41(3), 712-725.
- [52] Wei, Y., Feng, J., Liang, X., Cheng, M. M., Zhao, Y., & Yan, S. (2017). Object region mining with adversarial erasing: A simple classification to semantic segmentation approach. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 1568-1576).
- [53] Long, J., Shelhamer, E., & Darrell, T. (2015). Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 3431-3440).
- [54] Hakan Bilen, 2018 .Weakly supervised object detection [presentation]. Retrieved from <https://hbilen.github.io/wsl-cvpr18.github.io/assets/wsod.pdf>
- [55] Russakovsky, O., Lin, Y., Yu, K., & Fei-Fei, L. (2012, October). Object-centric spatial pooling for image classification. In *European conference on computer vision* (pp. 1-15). Springer, Berlin, Heidelberg.
- [56] Pdok. (2008, October 8).PDOK viewer. Retrieved from <http://www.pdok.nl/>. Accessed on March 2019. Datasets - PDOK, www.pdok.nl/.
- [57] Gupta, V. (2017, October 9).
Neuron
 . Modified from <https://www.learnopencv.com/understanding-feedforward-neural-networks/>
- [58] Pakhomov, D. (2016, November 22). Upsampling and Image Segmentation with Tensorflow and TF-Slim [Digital image]. Retrieved from <http://warspringwinds.github.io/tensorflow/tf-slim/2016/11/22/upsampling-and-image-segmentation-with-tensorflow-and-tf-slim/>
- [59] Benelux, SingularityU. “Carlo Van De Weijer on Real Intelligence.” YouTube, YouTube, 1 Sept. 2016, www.youtube.com/watch?v=I6sWZMR9OZM&t=32s.
- [60] Gomes, Lee. “Google’s Self-Driving Cars Still Face Many Obstacles.” MIT Technology Review, MIT Technology Review, 8 July 2016, www.technologyreview.com/s/530276/hidden-obstacles-for-googles-self-driving-cars/
- [61] Everingham, M., & Winn, J. (2011). The PASCAL visual object classes challenge 2012 (VOC2012) development kit. *Pattern Analysis, Statistical Modelling and Computational Learning*, Tech. Rep.

- [62] VImage Programming Guide. Performing Convolution Operations, Apple, 13 Sept. 2016, developer.apple.com/library/archive/documentation/Performance/Conceptual/vImage/ConvolutionOperations/ConvolutionOperations.html.
- [63] LeCun, Y., Bottou, L., Bengio, Y., & Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11), 2278-2324.
- [64] Activation Function. (n.d.). Retrieved from <https://deeptai.org/machine-learning-glossary-and-terms/activation-function>
- [65] Han, J., & Moraga, C. (1995, June). The influence of the sigmoid function parameters on the speed of backpropagation learning. In *International Workshop on Artificial Neural Networks* (pp. 195-201). Springer, Berlin, Heidelberg.
- [66] Nair, V., & Hinton, G. E. (2010). Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th international conference on machine learning (ICML-10)* (pp. 807-814).
- [67] Goodfellow, Ian, Yoshua Bengio, and Aaron Courville. *Deep Learning*. 301.
- [68] C. (2005, July 14). Neuron_model [Digital image]. Retrieved April 30, 2019, from https://upload.wikimedia.org/wikipedia/commons/6/60/ArtificialNeuronModel_english.png
- [69] Liu, W., Wen, Y., Yu, Z., & Yang, M. (2016, June). Large-margin softmax loss for convolutional neural networks. In *ICML (Vol. 2, No. 3, p. 7)*.
- [70] Vaillant, R., Monrocq, C., & Le Cun, Y. (1994). Original approach for the localisation of objects in images. *IEE Proceedings-Vision, Image and Signal Processing*, 141(4), 245-250.
- [71] Larhmam. (2018, October). SVM-margin. Retrieved from https://upload.wikimedia.org/wikipedia/commons/thumb/7/72/SVM_margin.png/300px-SVM_margin.png
- [72] Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., ... & Berg, A. C. (2015). Imagenet large scale visual recognition challenge. *International journal of computer vision*, 115(3), 211-252.
- [73] Sharma, Avinash. "Understanding Activation Functions in Neural Networks." *Medium, The Theory Of Everything*, 30 Mar. 2017, medium.com/the-theory-of-everything/understanding-activation-functions-in-neural-networks-9491262884e0.
- [74] Rother, C., Kolmogorov, V., & Blake, A. (2004, August). Grabcut: Interactive foreground extraction using iterated graph cuts. In *ACM transactions on graphics (TOG)* (Vol. 23, No. 3, pp. 309-314). ACM.
- [75] Borji, A., Sihite, D. N., & Itti, L. (2012). Quantitative analysis of human-model agreement in visual saliency modeling: A comparative study. *IEEE Transactions on Image Processing*, 22(1), 55-69.
- [76] Hu, P., & Ramanan, D. (2017). Finding tiny faces. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 951-959).

- [77] Kumar, J., Pillai, J., & Doermann, D. (2011, September). Document Image Classification and Labeling using Multiple Instance Learning. In 2011 International Conference on Document Analysis and Recognition (pp. 1059-1063). IEEE.

9 Appendix

9.1 Preliminary Research

To get some insights into the results we could achieve using labels from maps to label images, we first performed four experiments using an existing method. For this, we used Zhou peak response mapping method as it seemed to get good results, allowed for instance and semantic segmentation had a simple approach and had code publicly available.

9.1.1 A Single Roadside Object Class

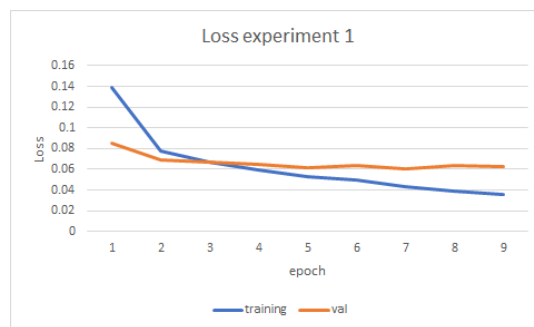
We started by adding one single roadside Class to the existing example model of Zhou et al; (13). For this object, we chose a street light as this is a very distinct object present in a large number of street view images. To allow this object to be added we modified the existing model to allow it to output an additional label i.e we changed to model from of 20 objects labels to 21 object labels.

These 20 pre-existing classes come from the 2012 VOC (61) training set. Which are also used to train the model, for the street light we added 500 training images and 100 validation images situated on street lights present in Utrecht. We ran training with the same parameters as Zhou et al. did.

For the creation of our training images, we used a FOV angle of 90 degrees and a distance range of 10 meters. As these parameters seem to allow for some noise in the measurements of the BGT maps and street view images.

When training we that the the validation set converges really quickly to a loss of 0.06, see Figure 54.

Figure 54: loss experiment 1



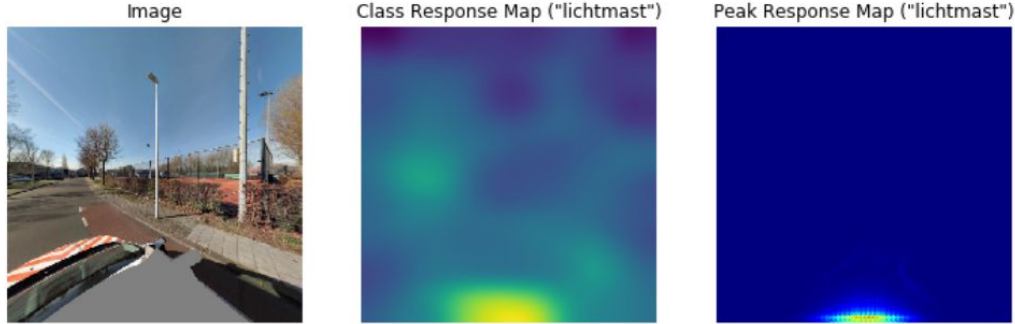


Figure 55: Class response map experiment 1 street light

When looking at our result, we find a classification F-score of 1.0 and an accuracy of 1.0 for the class street lights. However, when we then look at the CAMs of these street lights we find something different, see Figure 55.

Instead of finding the street lights it seems to focus on the car masks and a bit on the horizon. This result was expected as the car mask is clearly visible in every street light image. Furthermore, there are no negative examples also containing the car masks making it a unique feature for each image labeled as a street view image. Since this mask is clearly visible and occupies a large section of the images it becomes easily detectable resulting in the 100 % accuracy. This is actually not desirable as when we would include a street view image that doesn't include a street light it would still be labeled as such. To solve this we proposed to add more street view classes. This should then give negative examples for the horizon and the car present in the image, which would then force the network to look for the other objects present in the images.

9.1.2 Multiple Roadside Object Classes

For the second experiment, we added four more street view classes and removed the Imagenet classes present in our first experiment. Resulting in a dataset consisting of five different street view classes. The five street view classes are: Street lights, ABRI (the name for a shelter at a bus stop), Trees, Electrical utility cabinets, and garbage bins. These objects were chosen as they are a diverse set of differently sized objects. Furthermore, every class has 500 training images and 100 validation images resulting in a training set of 2500 images and a validation set of 500 images. It should be noted that every image is only labeled for one class and objects in the background of the classes are disregarded.

For this experiment we can see a clear difference in performance (see Table 31). Our model in this experiment seems to actually learn the appearance of the object (see Figure 56). Although there are still some issues with trees and street lights (see Figure 57)

We hypothesize this is because these objects are often present within the background of the other objects and since these other images are taken as negatives it is unable to fine-tune itself. This would explain our lower accuracy within these classes and also why it ignores certain versions of street lights.

confidence > 0.5	Sum	Street light	Abri	Garbage Can	Tree	Utility cabinet
true pos	381	59	83	86	72	81
false pos	96	12	6	27	25	26
false neg	119	41	17	14	28	19
true neg	1904	388	394	373	375	374
recall	0.76	0.59	0.83	0.86	0.72	0.81
precision	0.80	0.83	0.93	0.76	0.74	0.76
f-measure	0.78	0.69	0.88	0.81	0.73	0.78
accuracy	0.91	0.89	0.95	0.91	0.89	0.91

Table 31: Results on validation set with a confidence > 0.5

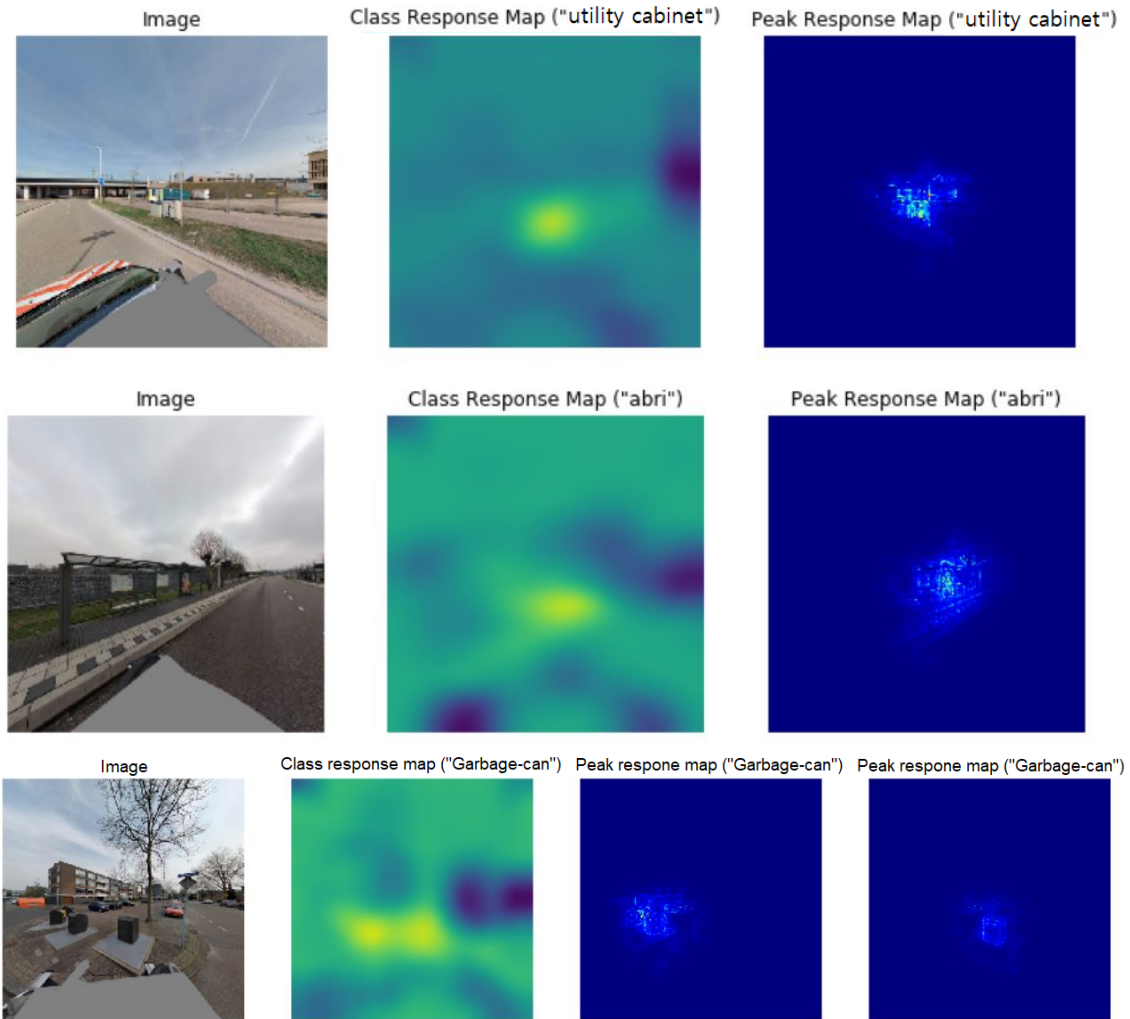


Figure 56: CAM able to localize objects

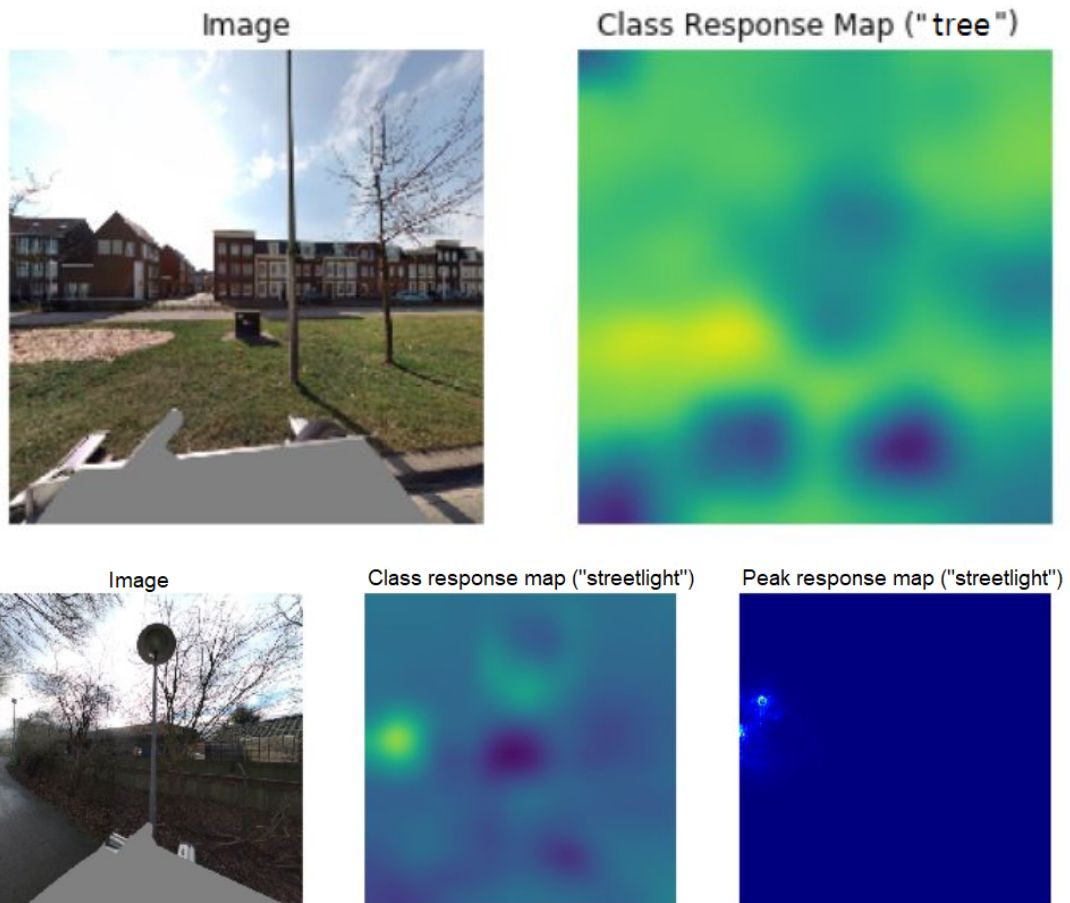


Figure 57: CAM not able to localize objects

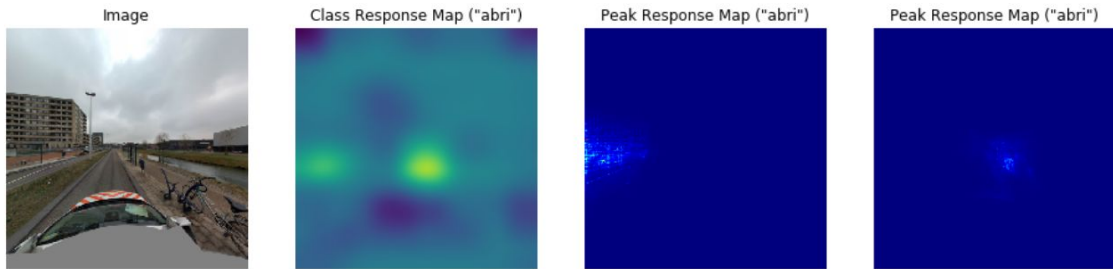


Figure 58: Finding multiple occurrences of ABRI's

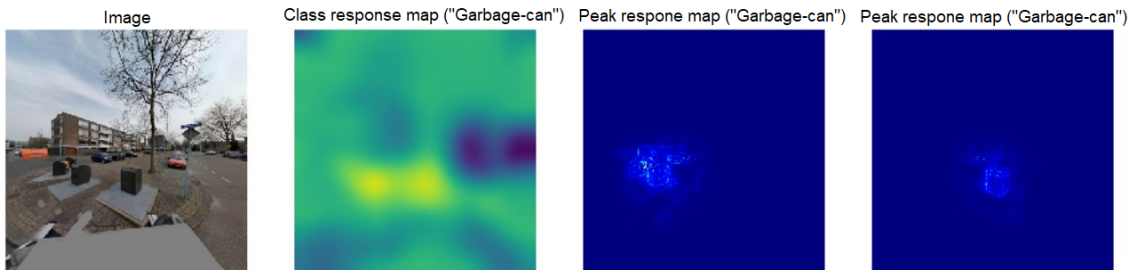


Figure 59: Finding multiple occurrences of Garbage cans

The other three objects, (Abri, Garbagecans, Utility cabinets), seem to do a lot better, even finding multiple of these objects within one image even though they only been trained on image-wide objects occurring per image, see figures 58, 59.

This supports our hypothesis as ABRI, Garbage-cans, and Utility cabinets are scarce objects compared to trees and street light meaning they are less likely to be in the background of other images where they would be used an incorrect negative example while training.

9.1.3 Adding Background Objects

In the third experiment, we proposed adding a view-cone to allow for the labeling of objects within the background of the image I.E. see figure 60.

This view-cone consists of two parameters a FOV (field of view) and a range. FOV corresponding to the horizontal FOV of the image and a range that determines the length of the cone. The view cone is then also split into two sections a closer section where map labels are seen as correct labels and a further section where labels present are seen as hard. This is added as the objects further away have a decreased possibility of being visible, e.g. a building or other objects might be blocking a direct line of sight to our object. We therefore label these objects as hard which the model then ignores when training.

To be able to compare our previous results to the results of this experiment we also use the validation set containing multiple labels on the model resulting from experiment 2.



Figure 60: Example of a view-cone, Smaller Cone indicates the labeled objects bigger Cone indicates all objects that are consider hard

For this experiment, we set the view cone FOV to the same FOV as the images I.E. 90° . The range of the view-cone is set at a range of 55 meters where after 20 meters labels are considered hard.

Our third experiment shows an dramatic decrease in performance for all objects (see Table 33). This is as the problem has greatly increased in difficulty as the model now has to predict multiple labels for each image. Furthermore, as we're taking into account background objects our classes in the 2500 images are no longer balanced. Certain objects are more often present within the background of images, see table 32. Explaining why trees and street lights still attain adequate performance. This large class imbalance could be downscaled a bit by using a weighted training I.E. when setting weights using the loss more prevalent objects are taking into account less to less prevalent objects. However, we still require a large set of negative images to be able to localize the correct objects which are a problem in the case of trees and street lights.

9.1.4 Balancing Datasets

The reduce the effect of the large class in balance we investigated used class weights to reduce they over-focused of certain classes over others. By weighting the classes we see a large increase in performance (see table 33). Furthermore, street lights are able to be found with the CAM showing that adding the view-cone helped in the ability to localize the object (see Figure 61).

	training	validation
Utility cabinets	527	108
Utility cabinets hard labels	35	6
Utility cabinets negative labels	1938	383
Street light	1322	280
Street light hard labels	458	85
Street light negative labels	720	133
GarbageCan	509	103
GarbageCan hard labels	3	1
GarbageCan negative labels	1988	388
Tree	1573	305
Tree hard labels	328	67
Tree negative labels	599	125
abri	532	104
abri hard labels	19	4
abri negative labels	1949	384

Table 32: number of images with certain label

Objects	Experiment 2	Experiment 3	Experiment 4
Street light	0.36	0.72	0.74
Abri	0.52	0.04	0.84
garbage Can	0.35	0.095	0.74
Tree	0.49	0.54	0.81
Utility Cabinet	0.66	0	0.68
Average	0.46	0.50	0.77

Table 33: Different performance of the different models on the multi label verification set.

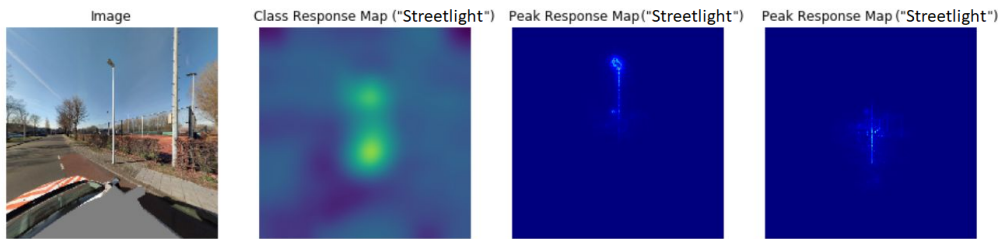


Figure 61: Street light are now able to be found by the CAM.

9.2 Resnet Versus VGGnet

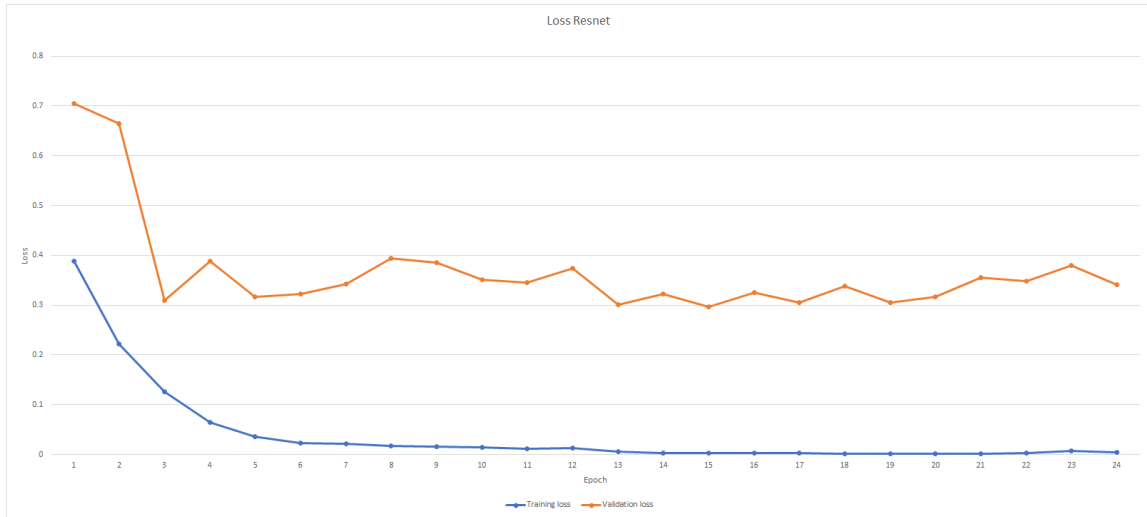
In this section, we describe our investigation in the performance of Resnet 50 versus the performance of VGGnet for the classification of objects within the automatically created street view dataset.

Within the PRM method proposed by Zhou et al (13) Resnet showed better performance than the VGGnet, we seek to investigate if these results hold when not using the PRM method but only the CAM method as proposed by Zhou et al; (34) on our dataset.

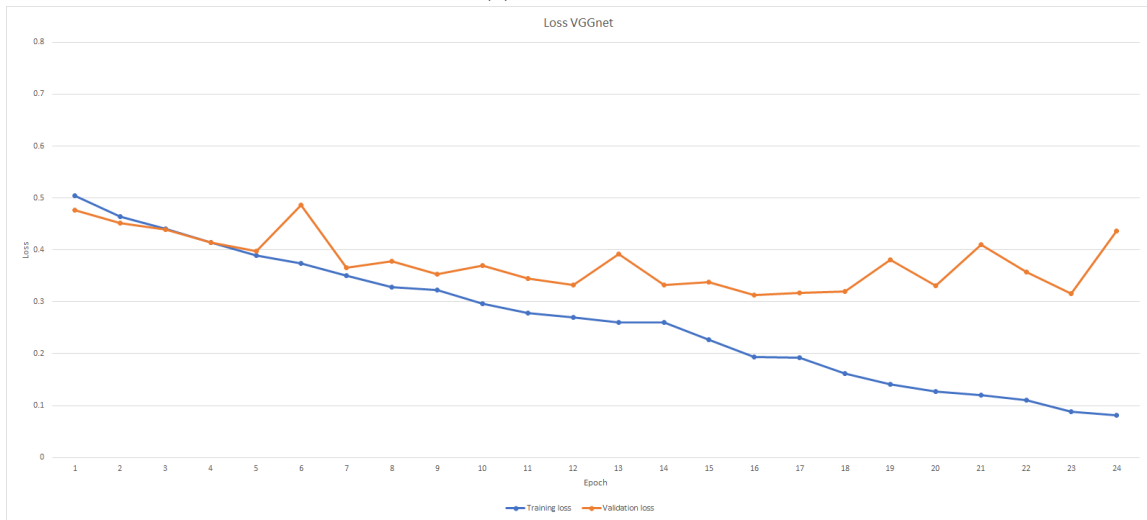
Both models were first pre-trained on Imagenet (44), after which the pre-trained models were trained on a small dataset consisting of 2500 images and 5 classes generated by our automatic training set creation method proposed in section 4.1. Using a batch size of 16 images, and with images sized 512 pixels by 512 pixels both models trained for 24 epochs. As the classes are unbalanced a weighted cross-entropy loss was used.

Our results show that both models loss seem to diverge relatively quickly as illustrated in Figure 62. VGGnet taking slightly longer than the Resnet 50 network. This is also reflected in the Fscores on the validation set per class as illustrated in Figure 63. Here we see VGGnet struggling for a longer time to learn to less prevalent classes requiring 17/18 epochs to learn the last two classes.

While the Resnet 50 network has a lower total average performance at the start of training as illustrated in Figure 64, this is mainly as the network learns all the different classes at the same time, where VGGnet instead first learn the most prevalent classes, I.E. Tree and Street light, resulting in a higher Fscore at the start of training. However, if we instead look at the average precision of all the classes we see our Resnet 50 network outperform the VGGnet network illustrated in Figure 65.



(a) Resnet 50 loss



(b) VGGnet loss

Figure 62: Loss of both networks

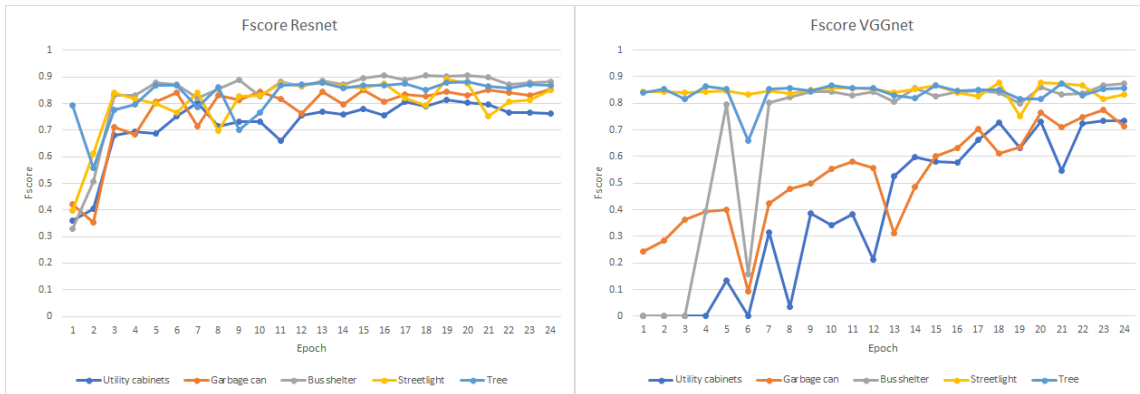


Figure 63: Fscores per class

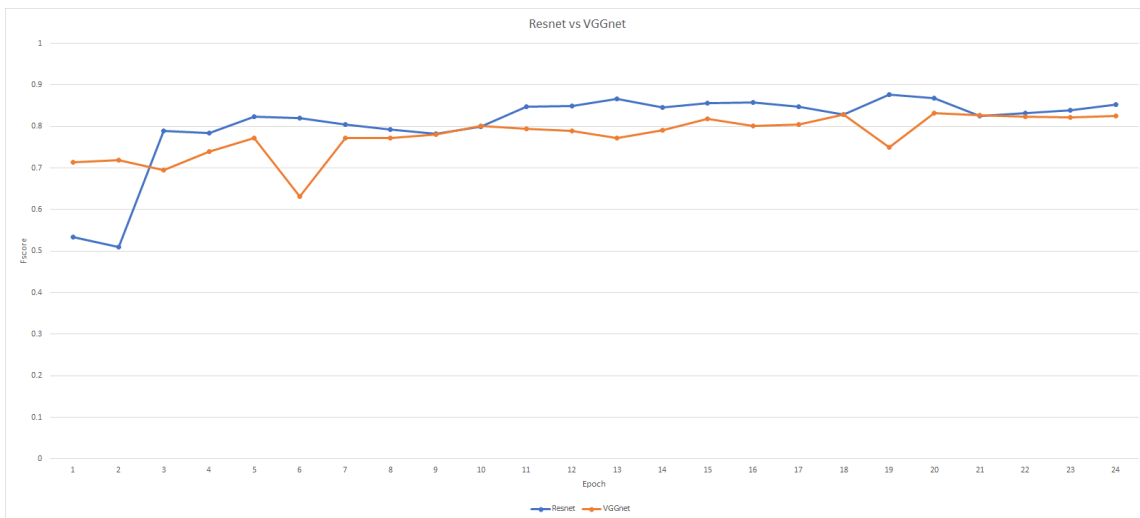


Figure 64: Average Fscore on validation set

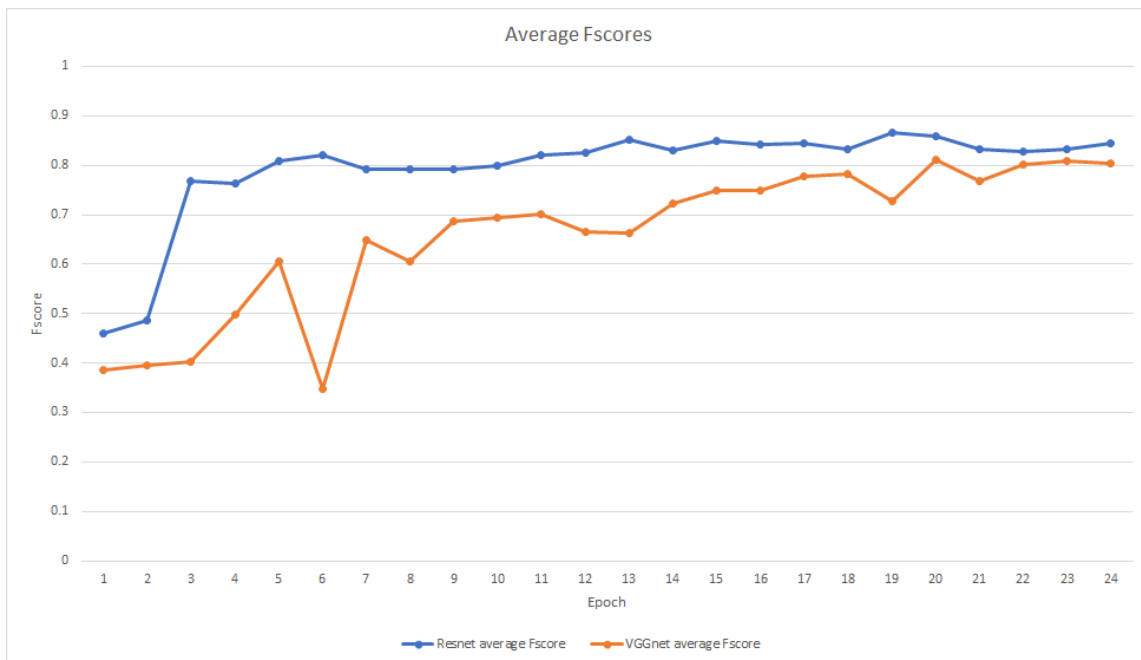


Figure 65: Fscores averaged on classes

9.3 Street View And Lidar Availability In The Netherlands



Figure 66: Dutch roads where street view images are available



Figure 67: Dutch roads where depth information images are available

9.4 Map Objects

Class name are represented by their dutch names as they are also labeled as such in maps.

9.4.1 DTB objects

Object	amount	Object	amount	Object	amount
hoogtepunt	1124343	reclamebord/zuil	3740	overstapconstructie geleiderail	410
boom	798639	reddingsmiddel	3363	snuffelpaal	341
verlichtingsobject	255984	verkeersbord met kabel	3183	verdrifpijl links	276
kolk	236510	huidspreker	3140	luchtoker	262
verkeersbord zonder kabel	144624	tv /camera	2811	meteorologisch instrument	217
meerpaal	106911	picknicktafel	2618	schoorsteen	192
hectometerpaal/bord	82356	navigatielicht	2325	gashouder	192
put (deksel)	79874	zonnepaneel	2048	windzak	179
bolder	55370	verdrifpijl rechts	2043	pompstation	178
kast/huis	54059	mast	1865	dukdaif	164
bermbeschermingsblok	31168	drukknoppaal	1753	watermolen	147
verfymbool	27876	brandblusser	1507	pijlmarkering rechtd/rechtsaf/linksaf	122
wegwijzer zonder kabel	25469	hoogspanningsmast	1323	pijl rechtsaf/linksaf	120
verkeerslicht	20160	mottobord	1320	radar	113
pijlmarkering rechtsaf	18430	zinkerbord zonder kabel	1255	pomp	102
dijkpaal	17468	aankondigingsbord met kabel	1124	molen	100
prullenbak/vuilnisbak	16711	pijlm rechtdoor/rechtsaf	1109	roodlichtcamera	97
matrixbord	16558	bewegingswerktuig	1091	uitwateringssluis	70
verklikker transportleiding	15606	ornament	1088	informatiebord praatpaal	66
bank(zit)	13990	bushalte	1087	dijk of kustwachttelefoon	57
trap	12512	brandkraan bovengronds	1037	zinkerbord met kabel	57
aankondigingsb zonder kabel	10874	pijlm rechtdoor/linksaf	996	tank	49
silo	10136	antenne	986	laadpaal	47
pijlmarkering rechtdoor	9995	praatpaal	807	schakelstation	47
pijlmarkering linksaf	8677	peilput	695	rioolput (zonder inlaat)	42
bolbaken	6998	peilschaal	671	schepenteller	5
pijler	6465	kabelbord zonder kabel	659	kabelbord met kabel	2
wegwijzer met kabel	4920	kilometerraai bord	641		
camera (algemeen)	4782	brandput	527		
naambord	4382	telefooncel	452		
meerstoel	4145	sein	432		
slagboom	3870	dynamisch route info paneel	417		

9.4.2 BGT-objects

Object	amount	Object	amount	Object	amount
kolk	2550088	abri	12654	dijkpaal	482
lichtmast	1983625	geleideconstructie	11953	zendmast	474
inspectie / rioolput	1266851	boomspiegel	9400	laagspanningsmast	418
afsluitpaal	509430	betaalautomaat	8404	scheepvaartbord	395
verkeersbord	229045	kunstobject	7881	drinkbak	359
verkeersbordpaal	197055	telecom kast	7634	fietsenkluis	340
brandkraan / put	170810	waarschuwingshenk	7051	debietmeter	336
waardeOnbekend	126200	haltepaal	6710	flitser	315
speelvoorziening	105821	bloembak	6494	hoogtemerk	313
fietsenrek	105265	druknoppaal	5554	telpaal	285
afvalbak	94533	zonnepaneel	5494	praatpaal	285
waterleidingput	87745	reclamebord	4952	zand/ zoutbak	197
bank	83727	openbare verlichtingkast	4486	sirene	189
afval apart plaats	52847	slagboom	4364	GMS sensor	180
CAI kast	50652	plaatsnaambord	4028	fontein	175
drainageput	49824	poller	4001	GMS kast	149
elektrikast	48237	pomp	3552	lichtcel	131
straatnaambord	48137	camera	3476	openbaar toilet	120
verkeersregelinstantiepaal	39584	reclamezuil	3445	rooster	67
meerpaal	36246	brievensbus	3254	radar detector	55
hectometerpaal	35872	portaal	3109	dynamische snelheidsindicator	54
wegwijzer	30758	picknicktafel	3024	wildrooster	36
lichtpunt	25130	grensmarkering	2764	weerstation	30
gasput	24015	waterstandmeter	2630	straalzender	24
bolder	20806	gaskast	2291	radarmast	24
informatiebord	18616	verkeersregelinstantiekast	2272	molgoot	20
container	18118	herdenkingsmonument	1630	windmeter	17
benzine / olieput	16756	telefooncel	1313	hoogtedetectieapparaat	6
rioolkast	13962	telkast	1188	balustrade	2
verklikker transportleiding	13681	betonning	875		
vlaggenmast	13655	parkeerbeugel	779		
bovenleidingmast	13028	wegmarkering	536		

9.5 Results

Loss Layer	Related CAM loss model		Equal CAM loss model		Label less CAM loss model		Depth less CAM loss model	
	convolutional	add	convolutional	add	convolutional	add	convolutional	add
Average	0.61	+0.03	0.59	-0.01	0.55	+0.08	0.6	+0.01
Traffic light	0.68	+0.06	0.7	0.0	0.65	+0.05	0.72	-0.02
Barrier post	0.59	+0.01	0.51	+0.01	0.43	+0.05	0.51	+0.05
Utility cabinet	0.55	-0.06	0.08	+0.03	0.11	+0.46	0.35	+0.06
Traffic sign	0.54	+0.05	0.56	+0.02	0.51	+0.05	0.55	+0.03
Manhole	0.41	+0.07	0.45	-0.06	0.43	-0.05	0.43	+0.05
Barrier/Gate	0.92	-0.02	0.95	+0.03	0.89	+0.03	0.95	-0.02
Street gully	0.09	+0.13	0.0	+0.03	0.0	+0.14	0.0	+0.06
Electronic freeway sign	0.83	0.0	0.86	+0.02	0.86	+0.06	0.88	-0.03
Bicycle stand	0.78	0.0	0.81	-0.03	0.73	+0.06	0.77	+0.02
Street name sign	0.55	+0.03	0.48	-0.13	0.0	+0.57	0.53	-0.09
Garbage Can	0.39	+0.1	0.25	+0.08	0.16	+0.36	0.38	+0.01
Street light	0.75	+0.02	0.76	-0.01	0.75	0.0	0.75	0.01
Camera	0.58	-0.07	0.38	+0.02	0.2	+0.35	0.41	+0.05
Bench	0.55	+0.04	0.55	-0.1	0.17	+0.37	0.6	-0.14
Roadside protection block	0.49	+0.11	0.14	-0.07	0.0	+0.58	0.3	-0.05
Information sign	0.31	+0.06	0.12	-0.02	0.0	+0.33	0.17	-0.0
Public play equipment	0.8	-0.04	0.73	-0.05	0.68	+0.13	0.73	+0.04
Hectometer post	0.24	-0.04	0.24	+0.06	0.33	-0.02	0.3	0.0
Ladder	0.52	-0.04	0.35	+0.05	0.53	+0.04	0.48	-0.02
Roadsign direction Arrow	0.72	+0.01	0.7	-0.03	0.62	+0.1	0.72	0.0

Table 34: Image-wide Classification using F-scores with the different CAM loss models showing the relative increase/decrease when using the 'add' layer versus the last convolutional layer, with a learning rate of 0.001.

Learning rate	0.01			0.005			0.001		
CAM Threshhold	0.3%	0.2%	0.1%	0.3%	0.2%	0.1%	0.3%	0.2%	0.1%
0.0	0.53	0.53	0.53	0.53	0.53	0.53	0.51	0.51	0.51
0.1	0.33	0.33	0.28	0.33	0.34	0.28	0.29	0.28	0.25
0.2	0.17	0.18	0.15	0.19	0.2	0.17	0.15	0.15	0.11
0.3	0.09	0.1	0.09	0.06	0.08	0.07	0.07	0.07	0.05
0.4	0.03	0.04	0.02	0.02	0.02	0.01	0.01	0.03	0.01
0.5	0.01	0.01	0.0	0.0	0.0	0.0	0.0	0.0	0.0
0.6	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
0.7	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
0.8	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
0.9	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0

Table 35: Comparison simple mask generated on NON CAM loss model add layer, Fscores at different IOU thresholds

Object	0.0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
Traffic light	0.64	0.3	0.18	0.09	0.03	0.0	0.0	0.0	0.0	0.0
Barrier post	0.58	0.08	0.02	0.0	0.0	0.0	0.0	0.0	0.0	0.0
Utility cabinet	0.57	0.36	0.36	0.15	0.1	0.0	0.0	0.0	0.0	0.0
Traffic sign	0.61	0.07	0.03	0.01	0.0	0.0	0.0	0.0	0.0	0.0
Manhole	0.48	0.23	0.19	0.15	0.08	0.08	0.04	0.0	0.0	0.0
Barrier/Gate	0.94	0.63	0.12	0.06	0.0	0.0	0.0	0.0	0.0	0.0
Street gully	0.3	0.3	0.22	0.17	0.17	0.13	0.04	0.0	0.0	0.0
Electronic freeway sign	0.81	0.4	0.16	0.08	0.04	0.04	0.0	0.0	0.0	0.0
Bicycle stand	0.29	0.29	0.29	0.29	0.29	0.0	0.0	0.0	0.0	0.0
Street name sign	0.08	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
Garbage Can	0.68	0.49	0.38	0.15	0.11	0.08	0.04	0.04	0.04	0.0
Street light	0.75	0.32	0.17	0.06	0.02	0.0	0.0	0.0	0.0	0.0
Camera	0.29	0.1	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
Bench	0.75	0.52	0.48	0.44	0.37	0.26	0.11	0.04	0.0	0.0
Roadside protection block	0.77	0.13	0.07	0.04	0.0	0.0	0.0	0.0	0.0	0.0
Information sign	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
Public play equipment	0.62	0.22	0.22	0.22	0.22	0.0	0.0	0.0	0.0	0.0
Hectometer post	0.42	0.32	0.21	0.11	0.0	0.0	0.0	0.0	0.0	0.0
Ladder	0.67	0.67	0.53	0.53	0.53	0.13	0.13	0.0	0.0	0.0
Roadsign direction Arrow	0.67	0.17	0.06	0.0	0.0	0.0	0.0	0.0	0.0	0.0

Table 36: The performance of the bounding simple method using the CAM generated by the Relative CAM loss model

IOU	0.0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
Average	0	+0.03	+0.02	+0.03	+0.04	+0.01	+0.01	+0.0	+0.0	0
Traffic light	+0.01	+0.05	+0.06	0	0	0	0	0	0	0
Barrier post	-0.09	-0.03	0	-0.02	0	0	0	0	0	0
Utility cabinet	+0.08	+0.09	+0.18	+0.06	+0.1	0	0	0	0	0
Traffic sign	-0.01	-0.06	-0.03	0	0	0	0	0	0	0
Manhole	-0.02	+0.01	+0.02	-0.02	-0.01	+0.08	+0.04	0	0	0
Barrier/Gate	0.01	+0.07	-0.07	+0.06	0	0	0	0	0	0
Street gully	+0.03	+0.12	+0.13	+0.08	+0.08	+0.08	0	0	0	0
Electronic freeway sign	+0.01	+0.27	+0.12	+0.04	+0.04	+0.04	0	0	0	0
Bicycle stand	+0.04	+0.04	+0.04	+0.04	+0.04	-0.25	0	0	0	0
Street name sign	+0.08	0	0	0	0	0	0	0	0	0
Garbage Can	-0.02	+0.1	+0.06	-0.09	+0.04	+0.01	-0.03	+0.04	+0.04	0
Street light	-0.02	-0.08	-0.01	+0.01	+0.01	-0.01	0	0	0	0
Camera	-0.08	0	0	0	0	0	0	0	0	0
Bench	0	+0.02	+0.05	+0.05	+0.12	+0.15	0	0	0	0
Roadside protection block	0	-0.03	-0.03	-0.03	-0.03	0	0	0	0	0
Information sign	-0.15	-0.15	-0.15	-0.15	-0.08	-0.08	0	0	0	0
Public play equipment	+0.28	-0.01	+0.01	+0.12	+0.22	0	0	0	0	0
Hectometer post	-0.27	-0.03	+0.04	+0.11	0	0	0	0	0	0
Ladder	+0.17	+0.17	+0.13	+0.33	+0.33	+0.13	+0.13	0	0	0
Roadsign direction Arrow	-0.08	+0.01	-0.05	-0.05	-0.05	0	0	0	0	0

Table 37: Performance increase when using the relative CAM loss method