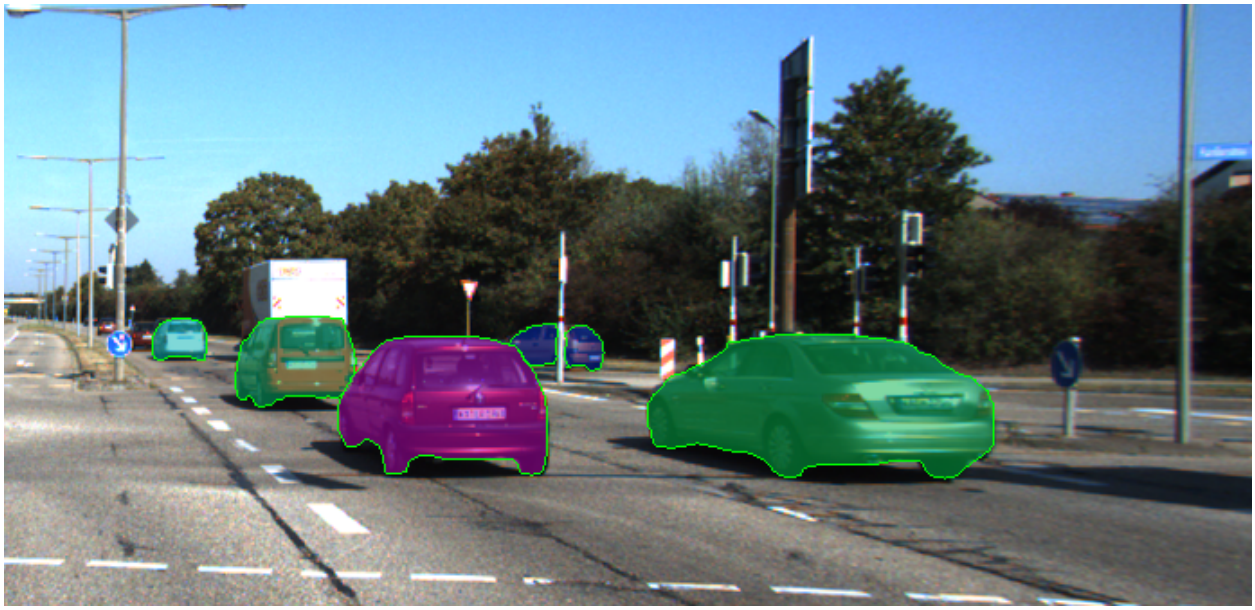
CHALMERS
UNIVERSITY OF TECHNOLOGY



Detailed Instance Segmentation of Cars in Sequences

Master's thesis in Systems, Control and Mechatronics

Tove Sohlman

Hanna Winblad

Department of Electrical Engineering
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2021

Master's thesis 2021

Detailed Instance Segmentation of Cars in Sequences

Tove Sohlman
Hanna Winblad

CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Electrical Engineering
Chalmers University of Technology
Gothenburg, Sweden 2021

Detailed Instance Segmentation of Cars in Sequences

Tove Sohlman
Hanna Winblad

Supervisor: Lennart Svensson, Department of Electrical Engineering
Advisors: Adam Brinkman & Isak Hjortgren, Annotell
Examiner: Lennart Svensson, Department of Electrical Engineering

Master's Thesis 2021
Department of Electrical Engineering
Chalmers University of Technology
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Cover: An annotated image from the KITTI MOTS [36] dataset generated by the network developed in this thesis.

Typeset in LaTeX
Gothenburg, Sweden 2021

Detailed Instance Segmentation of Cars in Sequences

Tove Sohlman
Hanna Winblad
Department of Electrical Engineering
Chalmers University of Technology

Abstract

Automated cars have tremendous potential to make transportation safer and more efficient for passengers. To realise this goal, the software controlling the vehicle must rely on machine learning algorithms, whose performance depends on the quality of the training data. In supervised learning, the data must be annotated with ground truth labels to be used for training. Today, this is mostly acquired by manually annotating the data, which is both expensive and inefficient. This is especially true for images that requires annotations on a pixel level in instance or semantic segmentation. It is common for images to belong to a video sequence where the images naturally have strong similarities that probably can be utilised to speed up the annotation.

This thesis investigates if sequential information in an image sequence can be used to automate the annotation process of segmentation further. Especially, the possibility of automatically receiving high-quality annotations for the frame at time t given manually annotated frames at time $t-1$ and $t+1$ is looked into. More specifically, a network architecture consisting of a deep convolutional neural network and a Transformer is used to segment one image given its two adjacent images, only using the annotation classes *car* and *background*. The main idea is that the deep convolutional neural network extracts feature maps from all the frames in the sequence, which are used to predict a first initial segmentation of the mask that is to be annotated automatically. The Transformer then uses this initial guess, together with the feature maps and the manually annotated masks at the adjacent frames, to adjust the boundary of the predicted mask.

The main results are that the Transformer improves the Jaccard index and Contour accuracy by 1.35 respectively 2.96 percentage points compared to segmentations produced by the deep convolutional neural network. The study also concludes that sequential information is of the essence for the Transformer to improve the segmentations. More explicitly, the feature maps, the manually annotated masks in the outer frames and the initial segmentation in the middle frame seem to play important roles to achieve high performance.

Keywords: Computer vision, instance segmentation, Multi-Object Tracking and Segmentation, Video Instance Segmentation, Transformer, deep convolutional neural network, deep learning, boundary key points

Acknowledgements

We would like to thank Annotell for the warm welcome and for showing great interest in our work and findings. Especially, big thanks to our advisors Adam Brinkman and Isak Hjortgren that has given us glorious support in understanding the problem and shared knowledge when troubles have occurred during the implementation.

We would also like to thank our examiner and academic supervisor Lennart Svensson, who has been generous with presenting fantastic ideas and guided us in the right direction throughout the project.

Tove Sohlman & Hanna Winblad, Gothenburg, June 2021

Contents

List of Figures

List of Tables

1
Introduction

The realisation of commercial self-driving cars would have a huge impact on society since it enables more efficient and safe transportation. When the vehicle is driving automatically, the passengers can focus on another task while being transported and hence more efficient transportation. The safety aspect of self-driving cars originates from the human errors which can occur in manually driven vehicles but are completely removed if the self-driving car works perfectly. To achieve this high standard, the car must rely, to a great extent, on machine learning algorithms and these algorithms need to be trained on data to work as intended. These algorithms can, for example, solve problems that occur when driving, such as how to best avoid an accident when an animal appears on the road unexpectedly.

A drawback with machine learning algorithms is their need for large amounts of training data. In supervised learning, this data requires ground truth labels to guide the training of the models. These labels can, for instance, represent the type of objects detected in the image, and the accuracy of the labels affects the performance of the algorithms. Today, the annotation of high-quality data on pixel-level is produced by annotators who manually identifies the object boundaries. This is very time-consuming, hence expensive, leaving great potential for savings if the level of automation is increased.

Two machine learning tasks that are important for the realisation of self-driving cars are semantic and instance segmentation, both of which treats annotation of images on a pixel-level. A problem in these tasks is that a high level of detail is needed to achieve accurate labels but also contextual understanding to identify the object class. The semantic segmentation task has been solved with neural networks in many attempts, such as DeepLabv3 [9]. Instance segmentation is the more difficult task of the two since it includes classifying each occurrence of an object class as an individual instance. This task has also been tackled using machine learning in attempts such as SpineNet [1], EfficientPS [2] and the Copy-Paste augmentation method used in [3]. All of the aforementioned models perform well, but if the networks are used to produce ground truth data, there is still room for improvements to achieve high-quality segmentations.

Another task, originating from instance segmentation of images, is instance segmentation of video sequences. A video sequence is a compound of many images occurring in a sequence, meaning that this task is similar to instance segmentation. The difference between the two problems is that segmentation of video sequences

additionally includes tracking instances over time. Therefore, this task is more complex since information needs to be connected between the frames. Further, similarities between frames belonging to the same video sequence usually are strong and can probably be used to improve segmentation quality. Attempts that have dealt with segmentation of video sequence are, among others, MaskTrack R-CNN [4], MaskProp [5] and TrackR-CNN [6]. None of these network structures utilise the sequential information to the full extent since it has not been used to improve the segmentation masks. Two attempts that use this information to produce the segmentation masks are VisTR [7] and VGCN [8], but the quality of these segmentations can still be improved.

This thesis investigates if machine learning models can utilise sequential information between frames in a video to increase the level of automation in the annotation process. Manually annotated segmentations will still be used to some extent in the developed network architecture. More specifically, a network that produces instance segmentation of cars in every second frame automatically, by utilising sequential information and manually annotated masks in the rest of the frames, is explored.

1.1 Objective

The vision for this thesis is to develop a method that utilises sequential information and some manual annotations to produce instance segmentation to be used as ground truth data for training and validation of machine learning models. This aim indicates that the masks produced by the network must be very accurate. Therefore, an acceptable outcome of the thesis is to evaluate if sequential information is beneficial for video instance segmentation and to what extent this information affects the quality of the segmentation masks.

To examine the evaluations mentioned above, the following questions must be answered:

- How can instance segmentation annotations be generated automatically in some frames by utilising already annotated frames that belong to the same sequence of images?
- What quality, according to some suitable metric, will the automatically generated annotations have? More specifically:
    - What quality will the automatically generated annotations have compared to a benchmark method?
    - What quality will the automatically generated annotations in the middle frame have when both of the adjacent frames, only one of the adjacent frames or none of the adjacent frames are manually annotated?

The benchmark method used is the segmentation network DeepLabv3 [9], which is described further in Section 2.2.1. The Jaccard index, developed by Jaccard [10], is the metric used for the evaluation of the segmentation masks, complimented with the Contour accuracy presented in [11] that evaluates the boundary. These metrics are explained in detail in Section 2.8.

1.2 Scope

The aim of the thesis is to achieve instance segmentation on a video sequence by leveraging both sequential information and manually annotated masks, where the latter is different compared to many other attempts to solve the same task. Further, this project only considers the segmentation of cars in traffic. An open-source dataset is used, both when training the final network structure and as the manually annotated masks that are input to the network. This means that the chosen dataset limits the project and transfer learning is desired when possible. Open-source scripts are used as the starting point for the project and are used to a great extent to complete the project.

Instance segmentation in sequence includes both segmentation and tracking of instances. In this project, the main aim is to produce segmentation masks of high quality rather than tracking the instances over the sequence. This can be explained by the possibility to use manual input and tracking is easy to solve manually. Therefore, bounding boxes around each instance is also provided to the network as a manual input, which reduces the task to semantic segmentation. Further, the results are not easily compared with other state-of-the-art networks since a Multi-Object Tracking and Segmentation dataset is used, but neither the tracking nor the object detection is considered.

1.3 Contributions

The main contribution of this thesis is a network structure that utilises information between frames in a video sequence, along with some manual input to produce the segmentation masks for the intermediate frame. More explicitly, one frame is annotated at a time by the network. The network utilises manually annotated masks in the two adjacent frames together with features extracted by a deep convolutional neural network for all three frames. Apart from the deep convolutional neural network, the architecture consists of a Transformer [14] that utilises the sequential information. The developed structure is inspired by recent research papers such as DeepLabv3 [9], DETR [16], PolyTransform [13], VGCN [8] and VisTR [7], but differs from all of them since the network utilises and, as shown from the results, benefits from the manual masks.

The developed network architecture receives a Jaccard index of 92.65% and a Contour accuracy of 92.78% on the test dataset, and therefore, is seen as a successful method for segmentation in sequence. Further, it is seen that the network leverages the manually annotated masks and that the quality of the produced masks is reduced when decreasing the number of manual masks available to the network. Therefore, it can be concluded that sequential information, in the form of manual masks, is beneficial for segmentation tasks.

1.4 Related work

Today the annotation process for instance segmentation is mostly done manually, which is very time-consuming. Since the frames in a video sequence are sampled at high frequency, similarities between frames are usually large. By annotating each frame manually and individually, information in adjacent frames is not utilised, and the annotator has to start all over when producing the next segmentation mask in the sequence. Therefore, it is reasonable to solve this problem using neural networks that aim to utilise the sequential information in the sequence to speed up the process. Previous works of segmentation networks operating at one frame at a time are of interest in this thesis, as well as algorithms utilising sequential information.

The task of producing instance segmentation for individual images has been tackled in several ways. Instance segmentation differs from semantic segmentation in that objects in the same class category, additionally should be labelled with separate instance identities. Many of the state-of-the-art solutions, such as SpineNet [1], EfficientPS [2] and the Copy-Paste augmentation method used in [3], build upon the Mask R-CNN network [12]. Mask R-CNN operates by finding candidate objects for an image, described by class labels, bounding box offsets and object mask, and from this decides which objects being instances by evaluating the object alignments. All of the mentioned instance segmentation networks shows promising performance but can still be improved, especially if the segmentations should be used as ground truth data. Further, none of the networks utilises sequential information, meaning that they treat each image separately, and therefore, are not possible to use directly in the project. Though, since the task of instance segmentation of video sequences is very similar to the tasks solved by these networks, they are used as inspiration.

One of the challenges to handle in the segmentation tasks is to achieve detailed segmentation for instances in different scales, which is especially difficult on the boundary of the objects. DeepLabv3 [9] is a semantic segmentation network that tries to solve the resolution problem that occurs from deep convolutional neural networks that downsample the feature map to a much lower resolution than the output segmentation by utilising atrous convolution. PolyTransform [13] is a network that attacks the instance segmentation task from a different perspective by focusing on the boundary. First, PolyTransform predicts masks for all instances in an image, whose boundaries further are described by polygons that are adjusted to align better with the object boundaries. Both of these ideas are used as starting points when the network structure in this project is developed since they solve problems related to the detail of the segmentations. However, neither of the networks utilises sequential information, meaning that extensions are needed before the ideas are used in the project.

Another segmentation task is Video Instance Segmentation (VIS), which is to produce instance segmentation of all the frames in a video sequence including tracking of all the instances over the whole sequence. MaskTrack R-CNN is a method presented in [4] together with a dataset for the VIS task named YouTube-VIS. MaskTrack R-

CNN claims to be the first technique used to take on the VIS task and builds upon the instance segmentation network Mask R-CNN [12], extended with an additional network branch that handles the tracking of the instances over the frames. Hence, this paper is highly related to the thesis since it also aims to solve the VIS task, but with the main difference that in this project, manual input is provided to the network. Therefore, another network structure, which incorporates manual information, is developed in this project.

Another more recent attempt to solve the VIS task is presented in [5] where mask propagation is utilised in a network structure called MaskProp. The mask propagation branch in the network propagates the instance-specific features in one frame to the adjacent frames to identify if the same instances reappear in the other frames. The propagation is thus exclusively used to solve the tracking in the VIS task and not to improve the segmentation masks. The thesis aims to use information in the prior and subsequent frame to produce the segmentation mask in the current frame, including manual information. Therefore, the described network does not solve the problem with the same presumptions as the thesis intends. Using manual information hopefully results in higher accuracy of the segmentation masks.

Transformers have also been used to solve the VIS task. The Transformer was first introduced in [14], where its superiority in natural language processing is demonstrated. The Transformer network consists of an encoder and decoder, both including multi-head attention. This architecture is adopted in VisTR [7], which uses a Transformer to utilise the sequential information in a video clip, i.e. a few adjacent frames, to produce segmentation masks for each instance in all the frames in the clip.

Transformers have been shown useful for other image recognition tasks as well. More specifically, a Transformer called SETR used for semantic segmentation is presented in [15], which, in contrast to VisTR [7], does not use a backbone to extract features before the frame is used as an input to the encoder. Further, in SETR [15], each frame is divided into patches. All the patches in one frame are viewed as a sequence, making it somewhat similar to the VIS task. Another image task where Transformers can be used is object detection. In [16], a network called DETR is presented, which utilises a CNN together with a Transformer and a feed-forward network to solve the object detection task. Overall, Transformers seem useful not only in language processing but also in various image tasks. This makes a compelling argument for using a Transformer to solve the task presented in this thesis. However, the inclusion of manual information remains unsolved by the aforementioned Transformer networks, and this extension is applied to the Transformer to solve the task in the thesis.

Another task similar to VIS is Multi-Object Tracking and Segmentation (MOTS). The difference between MOTS and VIS is that an instance that disappears and then reappears in a later frame is initialised as a new instance in MOTS, while VIS aims to classify the instance with the same label as before. This task is also highly relevant for the thesis since sequential information is used to tackle the instance

segmentation problem. A network structure called TrackR-CNN developed for the MOTS task is presented in [6]. This algorithm is based upon the Mask R-CNN [12] and utilises 3D convolution to consider the sequential information. TrackR-CNN has an additional association head that is used to track the instances throughout the sequence. Like most other existing methods, TrackR-CNN does not use sequential information or manual input to improve the segmentation masks, which is the aim of this thesis. Based on this, the network is not employed in the proposed network structure.

The Volumetric Graph Convolutional Network (VGCN) [8] is another interesting approach to handle the MOTS task, that has extended the work in Curve-GCN [17] to incorporate sequential information. The VGCN gets a video sequence provided with bounding boxes at the outermost frames, which are interpolated into the intermediate frames. The object segmentation in the bounding box is then described by control points along the boundaries, which are to be adjusted by pixel features through a Graph Convolutional Network (GCN). In VGCN, additional features, other than appearance features, are used to give the GCN more information about how the control points should be moved. One of these additional features is the movement of the pixels, i.e. optical flow, which is acquired using FlowNet2.0 [18]. The VGCN network utilises information between frames and some manual input in terms of bounding boxes in some frames. This structure, therefore, has many of the requested presumptions, but not all, since manual masks are not given as an input to the network. The network structure in the project has strong similarities with the VGCN network, but additionally incorporates the aforementioned manual information to receive high-quality annotations.

The earlier mentioned PolyTransform [13] has many similarities to VGCN [8] and Curve-GCN[17], in terms of that the methods focus on object boundaries and the possibility to utilise manual annotator input. In PolyTransform, the initial boundary guess is based upon a predicted instance mask whilst VGCN and Curve-GCN produce an ellipse centred in the bounding box. This means that the PolyTransform has a much more relevant starting point when adjusting the boundary than the other two methods. Further, PolyTransform employs the self-attending block from [14] to predict the offset from the guessed boundary. This seems like a reasonable strategy since the Transformer has a great advantage in connecting long-range dependencies compared to the GCN and is therefore also used in the thesis. Further, in this thesis, the initial guess of the polygon is acquired similarly to PolyTransform. The methodology for utilising sequential information is inspired by VGCN, with the GCN replaced with a Transformer following the structure of VisTR [7].

1.5 Thesis outline

The theory behind the methods and models used in the thesis are introduced in Chapter 2. Further, the methods applied in the thesis, such as the implemented network structure and dataset used, are described in Chapter 3. Chapter 4 presents the results from the experiments that were performed to evaluate and improve the

network structure. The results and methodology are discussed in Chapter 5. Finally, the thesis is wrapped up in Chapter 6 where also suggestion for future studies are proposed.

2

Theory

In this chapter the theory needed to understand the problem and solution presented in the thesis is explained. In Section 2.1, the segmentation task is defined, both in the standard case and when the frames appear in sequences. The task of detecting object boundaries is described in the same section. Section 2.2 introduces convolutional neural networks and describes one such network called DeepLabv3. The Transformer model and how this model can be used for image recognition are explained in Section 2.3. Section 2.4 mentions key point representations for polygons. The resizing methods binary interpolation and nearest neighbour interpolation are described in Section 2.5. Section 2.6 presents the Cross-Entropy loss, the Chamfer distance loss, and Deviation loss, followed by an explanation of boundary smoothing with Gaussian filtering in Section 2.7. Further, Section 2.8 describes the Confusion matrix, Jaccard index and Contour accuracy useful for evaluation.

2.1 Image segmentation

An image is a two-dimensional grid of pixels that together creates a context. Each pixel is displayed by colour intensity, commonly with the colour combination red, green and blue (RGB). Usually, the intensity of each colour is described to the computer with a value in the range $[0, 255]$ in a third dimension, often referred channel. The objective of the image segmentation tasks is to decide what class each pixel in the image belongs to based on these values.

When semantic segmentation is performed, a pre-determined number of classes is defined, and each pixel should be assigned to one of these classes. The classes can be both objects and other regions such as *sky* or *grass*. In the task of instance segmentation, each individual object is additionally given an instance ID, meaning that for example, two cars in the same image should be separable by the labelling, leading to that this task includes both segmentation and object detection. In instance segmentation, the classes are explicitly objects, and all other regions are assigned to the *background* class. Panoptic segmentation is a mixture of semantic segmentation and instance segmentation, where each object is given individual instance IDs and there exist additional background classes for regions that are not objects. An example of instance and semantic segmentation masks are illustrated in figure 2.1.

(a) (b)

Figure 2.1: Masks from the Cityscapes dataset [38] produced with semantic (a) and instance (b) segmentation with the classes background and car. Different cars are segmented as separate objects and illustrated with various colours when instance segmentation is applied, while all cars have the same label and colour for the semantic segmentation.

The resulting output produced from segmentation is a bitmap, where each bit corresponds to a pixel in the input image that holds an integer corresponding to the class or instance ID, depending on the task. The region of bits labelled as an object is often referred to as a mask. Since each pixel in an image has to be labelled, producing dense segmentation is challenging. Contextual information is usually extracted in a lower resolution than the original resolution and, therefore, it is difficult to restore the pixel-wise detail. Especially, small objects, occlusion and the area around object boundaries are problems that segmentation networks often struggle with.

An approach used in [17], [8] and [13] to solve the segmentation task is to only detect object boundaries in the mask. Once the object boundary is found, the region it surrounds can be converted to object segmentation. The boundary can be described by a polygon, which is the points creating the boundary if connected. Since the boundary is a problematic area when producing segmentations, this way of tackling the problem seems promising.

Two extensions of the segmentation task are Multi-Object Tracking and Segmentation (MOTS) and Video Instance Segmentation (VIS), which both includes instance segmentation of objects in a video sequence. The objective is to track each instance over time which is of importance for several automatic drive applications. In MOTS, an object that disappears is given a new instance ID if it reappears in subsequent images, while in VIS, the algorithm has to connect the instance ID along the whole sequence. The focus in these tasks is both to produce detailed segmentations and to connect the segmentations between frames. The tracking between frames is often solved by predicting how likely two masks in two images are to describe the same object, which indicates that there exist connections in the spatial and visual appearance of objects in adjacent images.

2.2 Convolutional neural networks

A convolutional neural network (CNN) has one or multiple layers utilising convolution to calculate the output from the layer, according to [19]. The learnable parameters in this layer are the weights that are used in the convolution. Usually these weights are called the filter. If the input to the convolutional layer I is two-dimensional, the filter K is two-dimensional and if the filter is not flipped, the convolution performed by the layer is as following according to [19]

$$S(i,j) = \sum_m \sum_n I(i+m, j+n) K(m,n) \tag{2.1}$$

where S is the produced output, referred to as the feature map. It is common for the filter to be smaller than the input, resulting in that every part of the input does not contribute to every part of the feature map. The filter can also move at different pace in each direction. The pace is called the stride and if the stride is larger than one the input is downsampled in the spatial dimensions by the convolutional layer. The output of the convolutional layer has a smaller width than the input. If this is not desirable, zero padding is commonly used to make the input larger.

It is common for the input to have three dimensions, for example, if the input is an image with colours. Usually, the depth of the third dimension is referred to as the number of channels in the input. According to [20], the filter has to have as many channels as the input since one channel of the filter, called the kernel, is used in the convolution at one channel of the input. The results from all the convolutions from all the kernels are added together to form the final output from the filter, as illustrated in figure 2.2. This means that even though the input is three dimensional, the output is only one dimensional. It is possible to produce a three-dimensional output by using multiple filters. The output from each filter then is concatenated to form the final output from the convolutional layer, meaning that the number of filters used in the layer decides how many channels the output will have.

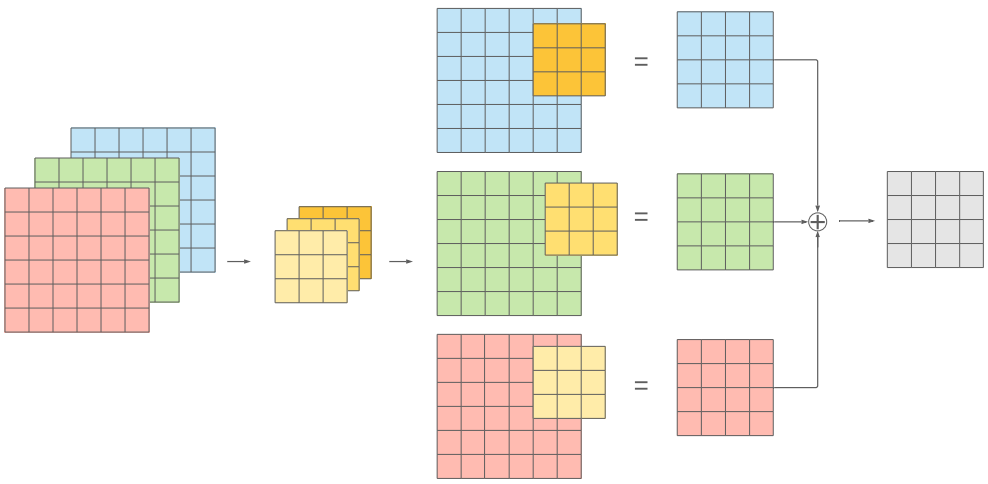


Figure 2.2: A $3 \times 3 \times 3$ kernel applied to a $6 \times 6 \times 3$ image, resulting in a $4 \times 4 \times 1$ output.

Usually, the output from the convolution is fed into a nonlinear activation function and then, the result of this operation is fed into a pooling layer, according to [19]. A pooling layer makes the network more resistant to translations in the input, i.e. if some features in the input are moved a few pixels, the output from the convolutional layer is almost not affected. This property is achieved by the pooling operator, which summarises the information from the input by looking at a few positions at a time and compile their information using some function decided by the type of pooling used. This function can, for example, be to set the value of the output to the maximum value of the values at the positions which the operator currently is processing.

2.2.1 DeepLabv3

DeepLabv3 [9] is a deep convolutional neural network utilised for semantic segmentation. A common problem in segmentation is that the feature maps, which are used for the classification of the pixels, need to have the same resolution as the input image, but the deep convolutional neural networks downsample the spatial resolution a lot to find the feature vectors. DeepLabv3 uses a feature extraction network that tackles this problem and the difficult scenario where objects appear in various sizes by utilising atrous convolution. Atrous convolution is a type of convolution where some of the kernel weights are set to zero in the spatial dimensions according to a constant rate, as in figure 2.3. This type of convolution makes it possible to decide the size of the neighbourhood, to which the convolution is applied to, without having to increase the number of parameters to be learned in the filter. Also, it is possible to capture information from parts in the image that is spatially far away without having to downsample as much as before, resulting in a higher resolution feature map.

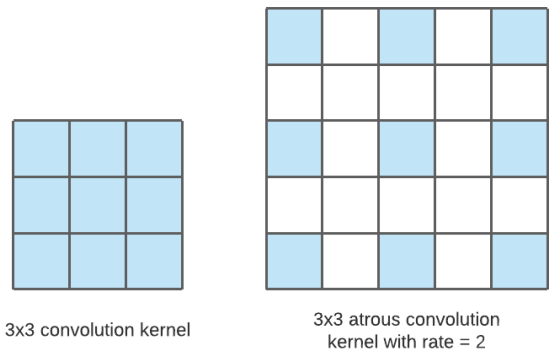


Figure 2.3: Two kernels, one used for convolution and one for atrous convolution. The convolution kernel has weights in all places, while the kernel used in atrous convolution has some of the weights set to 0.

The DeepLabv3 [9] model leverages ResNet [21] with the modification that atrous convolution blocks are used at the end of the network. DeepLabv3 also uses Atrous Spatial Pyramid Pooling (ASPP), which helps the network to tackle images where

the objects appear in different scales. ASPP was first introduced in [22] and this method consists of performing several atrous convolutions at different rates, modifying the output from each convolution separately and then concatenate the results. In DeepLabv3 [9], three parallel atrous 3x3 convolutions are performed in the ASPP branch, with rates 6, 12 and 18 together with a 1x1 convolution of the feature map. Also, another branch performs global average pooling and 1x1 convolution at the same feature map, parallel to the ASPP branch. The feature map from this branch is upsampled using bilinear interpolation and then concatenated with all the outputs from the ASPP branch. Then, the concatenated feature map is fed through a 1x1 convolutional layer. DeepLabv3 uses 256 filters in all the convolutions together with batch normalisation. All of this is seen in figure 2.4.

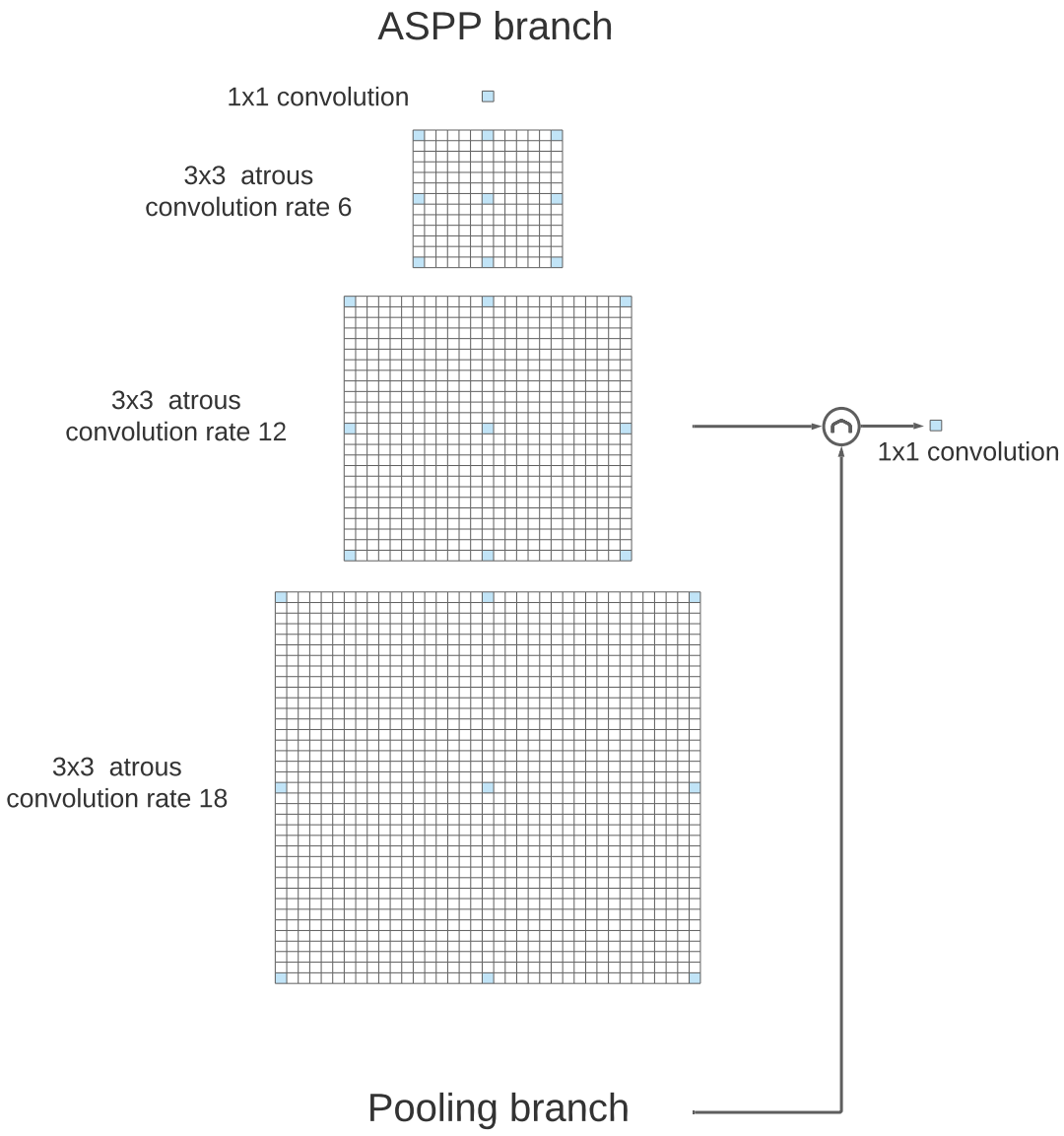


Figure 2.4: The ASPP branch and the Pooling branch used in DeepLabv3[9].

2.3 Transformers

The Transformer [14] builds upon the attention mechanism and consists of an encoder and decoder module, where both modules include multi-head attention layers. When using attention, the idea is to train the network to understand which parts of the input sequence that have dependencies. This mechanism is useful in translation tasks where each word in a sentence depends more on some words than others. It is also useful in image recognition tasks where multiple pixels, that not are necessarily spatially close, are needed to create a context.

The inputs to the multi-head attention module are called keys K, queries Q and values V. The interpretation of these inputs is quite abstract but is explained in [25] as finding similarities between each part of the sequence that is represented by the queries and the whole input that is represented by the keys. These similarities are the attention-weights that is further multiplied with the corresponding values in the input. Doing this successfully, a new and more expressive representation of the sequence that holds useful contextual information is produced.

Technically, the keys, queries and values are passed through three different linear layers for each of the attention heads. This is done to produce different keys, queries and values used as input to each of the attention heads. Each of the heads performs attention using the new keys, queries and values. This results in one output matrix for each of the heads, which further are concatenated and then fed into a linear layer to produce the final output from the multi-head attention layer. In [14], the attention mechanism is performed in each head using the following equation

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d}}\right)V \tag{2.2}$$

where d is the dimension of the queries and keys.

The overall architecture of the Transformer [14] can be seen in figure 2.5. Both the encoder and decoder consist of several sub-layers. The first sub-layer of the encoder is a multi-head attention layer, which processes the keys, queries and values that are the input to the encoder. The output from this sub-layer is added to the input to the encoder, and this sum is normalised. The normalised sum is fed to a feed-forward network before it is added to the normalised sum again. The resulting sum is then normalised once more to produce the final output from the encoder. This order of sub-layers is referred to as one encoder layer, and one Transformer can consist of multiple encoder layers after each other.

The output from the encoder is used by the decoder, which produces the final output from the Transformer. The decoder processes the previously produced output, which during training is assumed to be the ground truth regardless of the prediction from the decoder as explained in [23]. Masked multi-head attention is used as the first attention layer in the decoder since this enables efficient training of the decoder by calculating the whole output sequence from the decoder simultaneously,

according to [24]. This is possible since masked multi-head attention permits the attention mechanism to solely attend to the values that represent prior positions in the sequence than the position that is currently processed by the decoder, according to [14]. This also means that the whole correct output sequence can be fed to the decoder directly. The output from the masked multi-head attention layer is added to the decoder input and then normalised. This normalised sum is used as the queries to another multi-head attention layer. The keys and values used in this sub-layer are the output from the encoder. The output from this multi-head attention layer is added to the queries and normalised. After this follows a feed-forward layer and one last add and normalise layer. This stacking of sub-layers is referred to as one decoder layer, and one Transformer can consist of several decoder layers after each other.

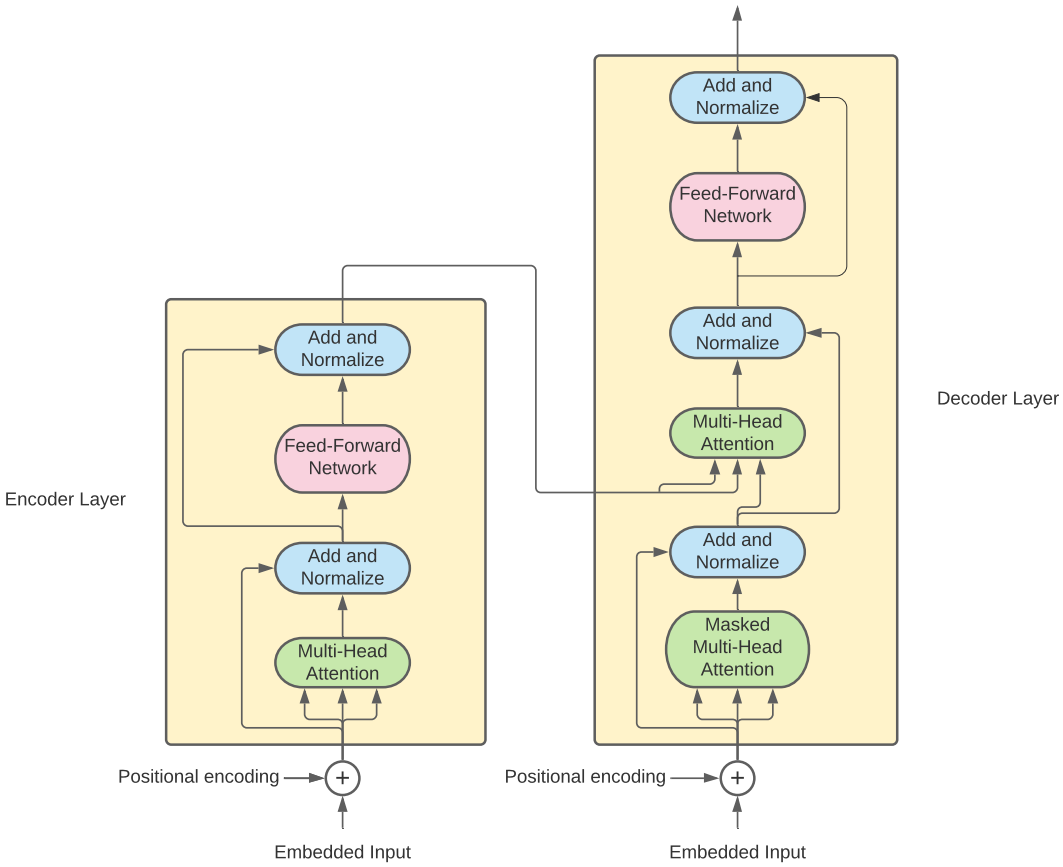


Figure 2.5: The Transformer [14] architecture consisting of an encoder and a decoder, including multi-head attention layers.

The Transformer module does not take the order of the sequence into account, meaning that this information has to be added to the input. The positional encoding can, for instance, be done as in [14], where the encoding is performed using sine and cosine waves, taking one value for each of the dimensions in the input. In [14], the positional encoding vector is added to the input, both in the encoder and decoder, as seen in figure 2.5.

2.3.1 Transformers for image recognition

The Transformer was invented for Natural Language Processing (NLP), for which it showed state-of-the-art performance [14]. Recently, the architecture has been adapted to several image recognition tasks, such as classification (ViT [26]), object detection (DETR [16]), segmentation (SETR [15]) and VIS (VisTR [7]). Utilising the Transformer architecture shows promising results in many of these tasks, which seems natural due to its possibility to extract contextual information. Still, it suffers from a quadratic increase in computational cost because of the attention mechanism when extending the input sequence.

The image recognition tasks differ from NLP since the decoder is allowed to attend to all other information in its input, i.e. the first layer in the decoder is exchanged from masked multi-head attention to regular multi-head attention. Another adjustment originating from the Transformer [14] that takes a one-dimensional input sequence, while images are two-dimensional, is that the two-dimensional images have to be reshaped before being fed to the network. SETR [15] and ViT [26] divide each image into patches where the pixels in a patch are further stacked in the channel dimension, resulting in that the sequence length is the number of patches. Both DETR [16] and VisTR [7] utilise a backbone network for feature extraction and put the height and the width dimensions of the feature map into one dimension, which becomes the sequence length. The positional encoding has to be converted from the one-dimensional case implemented in [14] to provide the Transformer with both spatial information and, in the case of sequential input, also temporal information. A modification of this was done for VisTR [7], where one-third of the channels were reserved for the positional encoding of each of the respective dimensions.

In classification, the need for details is not of the same importance as for the segmentation task since the whole image gets one label compared to one label per pixel. Hence, the resolution of the input for classification can be reduced, yielding a shorter input sequence, and therefore a lower computational complexity. However, for the segmentation task, high resolution is desirable, which makes the usage of the Transformer to become a problem of hardware optimisation.

Several attempts have been performed to reduce the increase in complexity with longer input sequences. One such trial is the sparse attention mechanism for longer sequences proposed in the NLP network BIGBIRD [27]. The main idea in BIGBIRD is that some of the positions, in the sequence, can attend and be attended by all of the other positions, and the rest of the positions only can attend to some neighbouring positions and some random positions. This sparse attention makes it possible to fit up to 8 times longer sequences at the same hardware. The image generation network Axial Transformer presented in [28] utilises the attention-mechanism alternately at positions in the x-dimension and y-dimension instead of attending all positions in the image simultaneously, leading to a reduction in computational complexity. An evolution from DETR [16] is the object detection network Deformable DETR [29], which replaces the computationally heavy Transformer encoder with a deformable attention module. The deformable attention module is a sparse attention module

that enables the input queries to the decoder to attend to a small set of relevant positions in the input feature map instead of the whole feature map.

2.4 Key point representation

As mentioned, the boundary of a segmentation mask can be represented by a polygon described by key points connected with straight lines. A motivation for finding and adjusting key points is that segmentation networks usually struggle the most on the object boundaries. Hence refinements here are of the essence. Two inspiring networks that apply this method are PolyTransform [13] and VGCN [17]. In VGCN, the initial polygon is an ellipse centred in the image, while PolyTransform extracts the key points from a first prediction of the mask using Suzuki's algorithm [33]. The number of key points used to describe the boundary could be either all of the boundary pixels or a smaller number yielding an approximation of the boundary. In OpenCV [34], Suzuki's algorithm is implemented with different kind of approximations of the key points available. In figure 2.6, the difference of a non approximated key point representation compared to a simple approximated representation is seen.

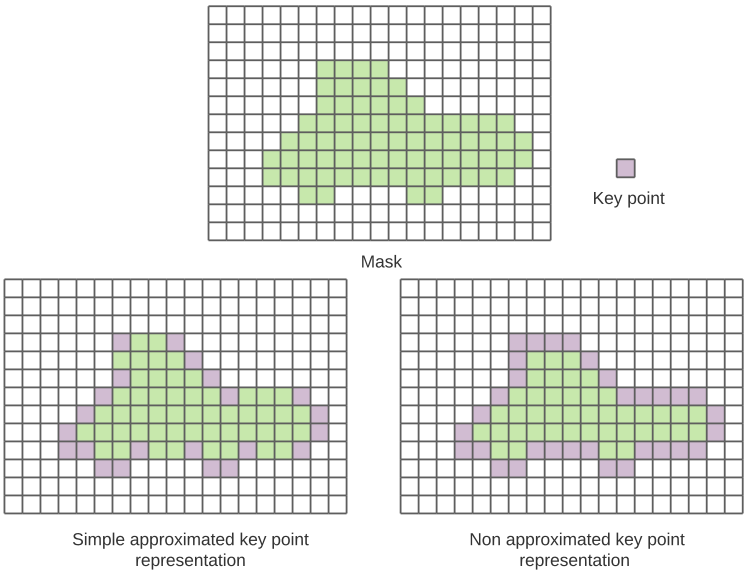


Figure 2.6: Two different ways to represent the boundary using key points. In the simple approximation representation, boundary pixels representing a straight line yield no key points. In the non-approximation representation, they do.

2.5 Resizing methods

An image can be resized by either making it smaller, called downsampling, or larger, called upsampling. Both up- and downsampling are procedures that use some function to produce a spatially larger or smaller output. The new pixel values in the new image are calculated from the old pixel values based on the earlier mentioned

function, which is decided by the type of resizing method used. One such common method is bilinear interpolation, which calculates the new pixel values by bilinearly interpolate the old pixel values. Another resizing method is nearest neighbour interpolation, which instead decides the new pixel value by letting it inherit the value of the old pixel that it is spatially closest to. An example of a binary image upsampled with nearest neighbour interpolation and bilinear interpolation is illustrated in figure 2.7. It is seen that in the binary case, the nearest neighbour interpolation method keeps the binary form of the input while bilinear interpolation does not.

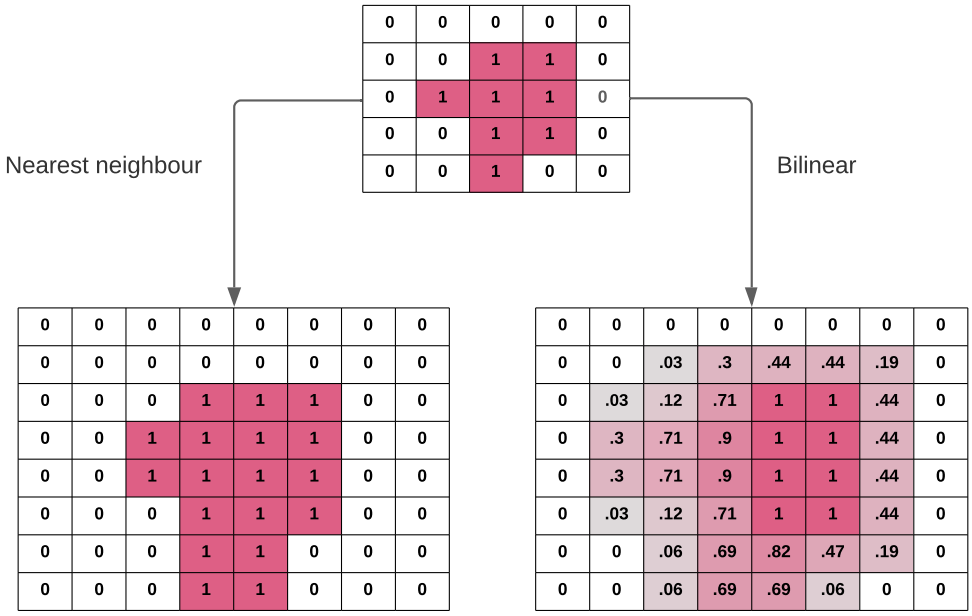


Figure 2.7: Upsampling of a binary mask using the methods nearest neighbour interpolation to the left and bilinear interpolation to the right. Nearest neighbour interpolation yields a binary mask, while bilinear interpolation can result in decimals for some pixels.

2.6 Loss functions

The learnable weights in a neural network need to be trained to solve their intended task. Gradient descent is usually used as the optimisation method to find the parameters best fitted for the network. For this, a function to minimise with respect to some parameters in the network has to be defined. This function is called a loss function, and the value of the loss is calculated for each input in the batch separately. Further, the final loss value for the batch is computed as the mean value of all the losses in the same batch. A common loss function for classification tasks is the Cross-Entropy loss which is further explained in Section 2.6.1. Two loss functions that can be used when the input to the network is points are the Chamfer distance loss and the Deviation loss. These losses are described in Section 2.6.2 respectively

Section 2.6.3.

2.6.1 Cross-Entropy loss

The Cross-Entropy loss is commonly used for classification tasks. The idea with this loss is that a high value is generated if the prediction corresponds to the wrong class and a low value if the prediction corresponds to the right class. Hence it helps the parameters in the network to update in the direction yielding the largest amount of correct predictions.

In binary classification, meaning that it is only two classes the network should learn to distinguish between, the binary Cross-Entropy function is used. According to [30], this loss function is defined as following

$$L_{BCE}(y, \hat{y}) = -\left(y\log(\hat{y}) + (1-y)\log(1-\hat{y})\right) \tag{2.3}$$

where y is the ground truth class and $\hat{y}$ is the predicted probability for class 1. When the classification task is in the form of segmentation, i.e. classify all the pixels in an image, it is still common to use the Cross-Entropy loss. In this case, the loss must first be calculated per pixel and then taken an average of all the resulting losses to obtain the final loss for the segmentation of the image.

2.6.2 Chamfer distance loss

The Chamfer distance loss is used to train the network to minimise the distance between two point sets. This loss is utilised in the training of the instance segmentation network PolyTransform [13] and is defined in the same paper as following

$$L_C(P, Q) = \frac{1}{|P|}\sum_i \min_{q \in Q} \|p_i - q\|_2 + \frac{1}{|Q|}\sum_j \min_{p \in P} \|p - q_j\|_2 \tag{2.4}$$

where P is the set of predicted points and Q is the set of ground truth points.

The intuition behind the loss is that it is optimised when the two point sets are as close as possible two each other. Each predicted point is assigned to the closest ground truth point. This procedure is also done in a reversed manner, meaning that each ground truth point is assigned a predicted point based on the distance. The bi-directional assignment prevents all points in the predicted point set to align with the same ground truth point.

2.6.3 Deviation loss

PolyTransform [13] uses a Deviation loss as a complementary to the Chamfer distance loss, which prohibits the distances between key points in the polygon to deviate too much. The idea is that this loss favours that many points move small distances over few points moving much, which means that the predicted point set has to be

adjusted synchronised. The loss is expressed with the following equation

$$L_D = \sqrt{\frac{\sum(e - \bar{e})^2}{n}} \tag{2.5}$$

where $\bar{e}$ is the mean distance between the key points in the ground truth polygon, e is the distance between two adjacent key points in the polygon, and n is the number of key points in the predicted polygon.

2.7 Boundary smoothing

To acquire a more continuous representation of the boundary, smoothing can be applied to the resulting polygon. This can be done by a convolution kernel distributed as an approximation of a Gaussian filter [31]. The form of a one-dimensional Gaussian filter $f(x)$ is defined by form of a one-dimensional Gaussian filter $f(x)$ is defined by

$$f(x) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left(\frac{x^2}{2\sigma^2}\right) \tag{2.6}$$

where σ is the standard deviation of the Gaussian distribution. A one-dimensional kernel can be used with weights approximating the filter, which is applied to the boundary in the x-dimension and y-dimension separately. The kernel computes a new coordinate for each point at the polygon based on the neighbouring points to achieve a more continuous boundary. As for all filters, the standard deviation and the kernel size have to be tuned to fit the application.

2.8 Metrics

To enable evaluation of the performance of an algorithm and comparison between different solutions, metrics are essential. Each metric gives a quantitative measurement of some aspects, and it is hard to find a metric that covers all areas of interests in an evaluation. Hence, it is motivated to use more than one metric to include a wider spectrum of an algorithm's ability to accomplish a task. Even though quantitative metrics are good indicators of performance, visually analysing the results add additional valuable understanding of the network's overall performance.

Both the segmentation masks and the boundary precision are evaluated visually and with metrics. To find a trend in the performance in pixel accuracy between the classes, a confusion matrix as described in Section 2.8.1 is studied. The Jaccard index J is used to evaluate how much the segmentation mask overlaps with the ground truth mask, while the boundary is evaluated using the Contour accuracy F. These two metrics are described in Section 2.8.2 and Section 2.8.3 respectively.

2.8.1 Confusion matrix

A confusion matrix can be used to measure the performance of the network for the different classes. A high number is desired at the diagonal since these are the

number of pixels that are predicted as the ground truth. The off diagonals hold the number of pixels that are predicted to the wrong class. In the binary classification task, class 0 is often referred negative, and class 1 positive. A correctly labelled pixel is referred true and incorrectly labelled pixel false. This structure is illustrated in figure 2.8. The values in the boxes either can be the actual number of occurrences, the row-wise normalised numbers or the row-wise percentage.

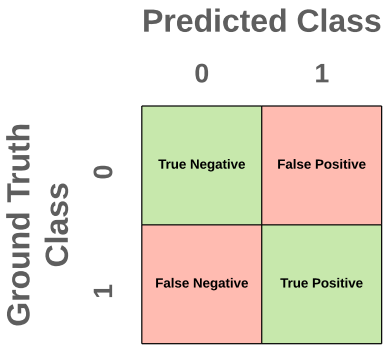


Figure 2.8: An illustration of the structure of the confusion matrix, which rows consist of the ground truth class and the columns of the predicted class.

2.8.2 Jaccard index

The Jaccard index J, developed by Jaccard [10] and also known as Intersection over Union (IoU) is defined by

$$J(M, G) = \frac{|M \cap G|}{|M \cup G|} \tag{2.7}$$

where for each sample, M are pixels labelled as the currently evaluated class in the segmentation produced by the model, G are pixels labelled as the currently evaluated class in the ground truth segmentation. An illustration of this metric is seen in figure 2.9.

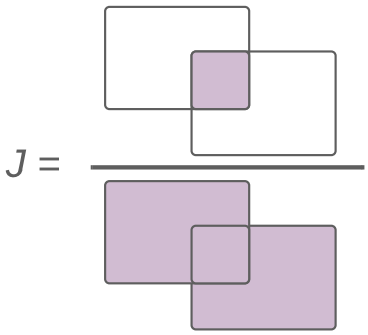


Figure 2.9: An illustration of the Jaccard index, where the intersection of two masks is divided by the union of the masks.

A high value of the Jaccard index is desirable and $0 \leq J(M, G) \leq 1$. The Jaccard index is calculated for all samples in the evaluation dataset, for each class separately and lastly, the mean over the dataset is taken for each class. This enables analysis of the network performance for different classes, and both false positives and false negatives are taken into account. False positives are pixels that are incorrectly categorised as the evaluated class, and false negatives are pixels that should be labelled as this class but are missed. It is common to use the Jaccard index in segmentation tasks, but since boundary pixels for big objects are a small proportion of the total amount of pixels covering the object, an improvement in boundary accuracy does not result in a significant increase of the Jaccard index. Due to the same reason, separating the metric into object scale to investigate how the network can handle small and big objects might improve the analysis even further.

2.8.3 Contour accuracy

The Contour accuracy F adopted for the segmentation task in [11], is a good compliment to the Jaccard index since it captures the network performance on the boundary segmentation. The metric is defined by

$$F(M, G) = \frac{2P_c R_c}{P_c + R_c} \tag{2.8}$$

where P_c and R_c are the precision and the recall of the predicted contour of the model compared to the ground truth contour. Precision is the fraction for how many of the pixels predicted to belong to the object boundary that belongs to the object boundary according to the ground truth. Recall is the fraction describing how many of the pixels that belong to the ground truth boundary are predicted correctly. Since a contour match at the pixel level is difficult to achieve, especially for large images, the contours can be thickened with binary dilation. If contour pixels are given value 1 and the rest of the pixels value 0, binary dilation ensures that all pixels in a fixed area, called a disc, centred around each pixel in the original contour takes value 1 [32]. A toy example of this is illustrated in figure 2.10.

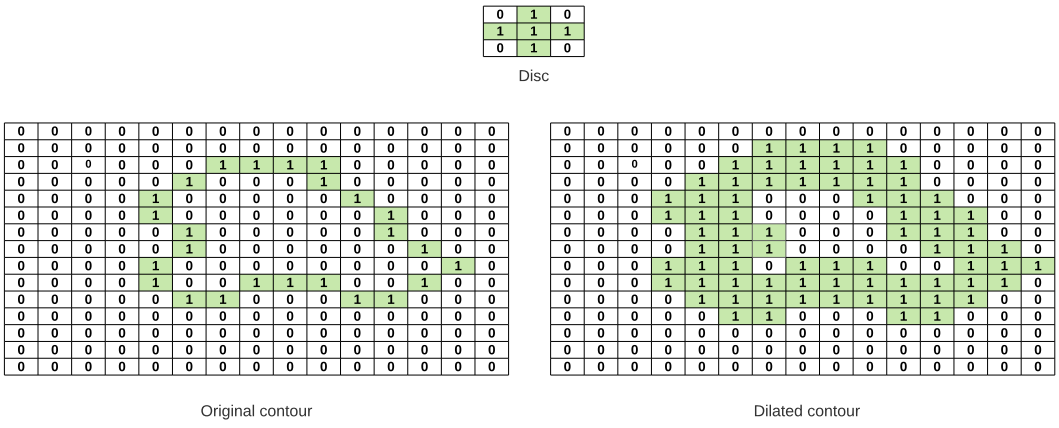


Figure 2.10: A toy example of binary dilation, where the original contour is shown to the left and the same contour dilated with the disc to the right.

3
Methods

This chapter presents the developed network structure in Section 3.1, followed by a description of the dataset used and how it was processed in Section 3.2. Lastly, the training and the evaluation setups are explained in Section 3.3.

3.1 Network architecture

The main idea behind the developed model is to achieve detailed instance segmentation in a frame by utilising the manually annotated masks in the two adjacent frames. The output is automatically generated masks in every second image, provided their manually annotated bounding boxes and manually annotated masks in the rest of the images. The developed network focuses on adjusting object boundaries since this is the most crucial and difficult part of the segmentation task.

The network structure is seen in figure 3.1. The network processes three frames at a time, i.e. the prior, current and subsequent frame. These are denoted frame at time $t-1$, t and $t+1$ throughout the report. The first step is to predict an initial guess of the mask in time t by employing a deep convolutional neural network. This part of the network is referred to as the backbone and explained in detail in Section 3.1.1.

A boundary representation is extracted from the backbone mask by finding key points that create a polygon with the object shape. These key points are further adjusted by a Transformer, described in Section 3.1.2. The Transformer's task is to use the manual masks at time $t-1$ and $t+1$ to update the mask at time t with details missed by the backbone. The Transformer needs additional information about all of the images to find the connections between the frames. Hence, the backbone also extracts feature maps for all three frames to concatenate with the masks to form the input of the Transformer.

3.1.1 Backbone

The purpose of the backbone is to find a first guess of the segmentation mask for the frame that is to be segmented. It is also used to elaborate the more informative way of expressing the images as feature maps. The deep neural network DeepLabv3 [9] with some minor modifications is employed, since DeepLabv3 has shown state-of-the-art result in the semantic segmentation task. This model is available with pre-trained weights in PyTorch [35]. The pre-trained network can distinguish be-

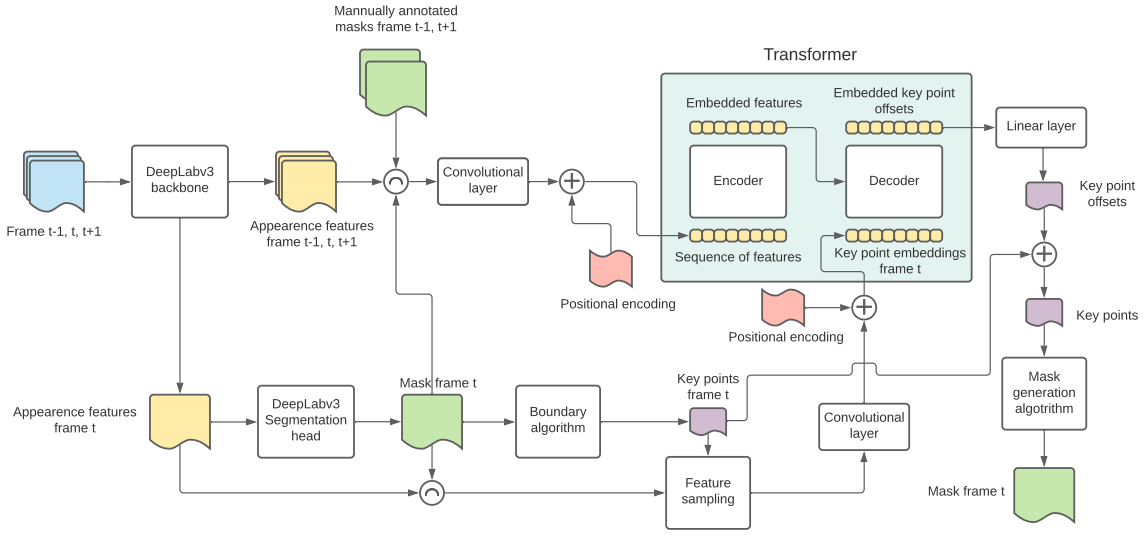


Figure 3.1: The overall network structure consisting of a backbone, a Transformer and a fully connected linear layer.

tween 21 classes, and hence the segmentation head is retrained to solely separate classes *car* and *background*.

The input to the backbone is achieved by the preprocessing of the data, which is further explained in Section 3.2. The original images are cropped to smaller images where for each image, one car is segmented and all other cars are labelled *background*. Hence, the task is reduced from instance segmentation to semantic segmentation, leading to that DeepLabv3 is possible to use even though it is a semantic segmentation network.

As mentioned, the input to DeepLabv3 is images cropped around the same instance in three consecutive frames $t-1$, t and $t+1$. These frames are processed by the backbone individually to produce their corresponding feature map. Since DeepLabv3 usually outputs a segmentation map, a modification is done to make it output a feature map as well. The spatial resolution of the extracted feature maps is eight times smaller than the input resolution. They are further resized by bilinear interpolation to achieve a feature vector for every pixel in the input resolution to the Transformer encoder. The number of channels in the feature vectors is 256 as the standard implementation of DeepLabv3.

The segmentation head of DeepLabv3 is utilised for the frame at time t, i.e. the frame that is to be annotated by the network. The head performs segmentation for this frame without making any hard predictions, except to evaluate the metrics and acquire the boundary polygon of the resulting segmentation mask. This means that the output from this step is a probability tensor with two channels, each describing what probability DeepLabv3 estimates that the pixels belong to class *background* or *car*. Only the channel describing the probabilities for class *car* is further utilised in

the network, and this tensor is referred to as the probability map throughout the report.

3.1.2 Transformer

A Transformer is employed to adjust the boundary polygon predicted by the backbone to better fit the true boundary. The Transformer uses attention to learn relationships between everything in the whole input sequence. Hence, this network architecture has a promising potential to perform well in the task of connecting both spatial and temporal information.

The same Transformer encoder-decoder structure with attention blocks as in DETR [16] is used in the thesis. This Transformer architecture is similar to the one presented in [14], except for the multi-head attention used in the decoder and the positional encoding. Masked multi-head attention is not needed in this thesis nor DETR since all the input embeddings can be processed in parallel, i.e. there is no need to have the previous output when producing the next output. Therefore the masked multi-head attention layer in the decoder is replaced with a multi-head self-attention layer. The positional encoding used in this work considers both spatial and temporal information. These encodings are produced using the same methodology as in VisTR [7], which uses a modified version of the positional encoding using sine and cosine waves in [14] fitted for this multidimensional case, with both spatial and temporal encoding. Further, the positional encoding is added to the queries and keys before all the attention layers in the Transformer, which differs from the implementation in [14].

The Transformer takes two inputs, one to the encoder and one to the decoder, where the latter input is referred to as input queries. The input to the encoder consists of the features with 256 channels extracted by the backbone for frames $t-1$, t and $t+1$. These three tensors are extended, then concatenated into one and then finally flattened. This procedure of extending and concatenating the feature maps is illustrated in figure 3.2 and described below.

The extension is done to give the Transformer even more information and consists of an extra channel, including the segmentation masks. This channel has values from zero to one that represents the prediction of the segmentation, where a value close to one represents a high probability of the pixel being *car* and zero corresponds to *background*. For the manually annotated frames, the added masks are the ground truth masks, while for the automatically annotated frames, the extra channel is the probability maps produced by DeepLabv3. The masks and probability map have to be resized to the desired input resolution of the Transformer since they are concatenated with the feature maps. The manually annotated masks are binary tensors and therefore are resized using nearest neighbour interpolation since this keeps their binary format. The probability maps for the frames at time t consist of decimal numbers between zero and one and are therefore resized with bilinear interpolation.

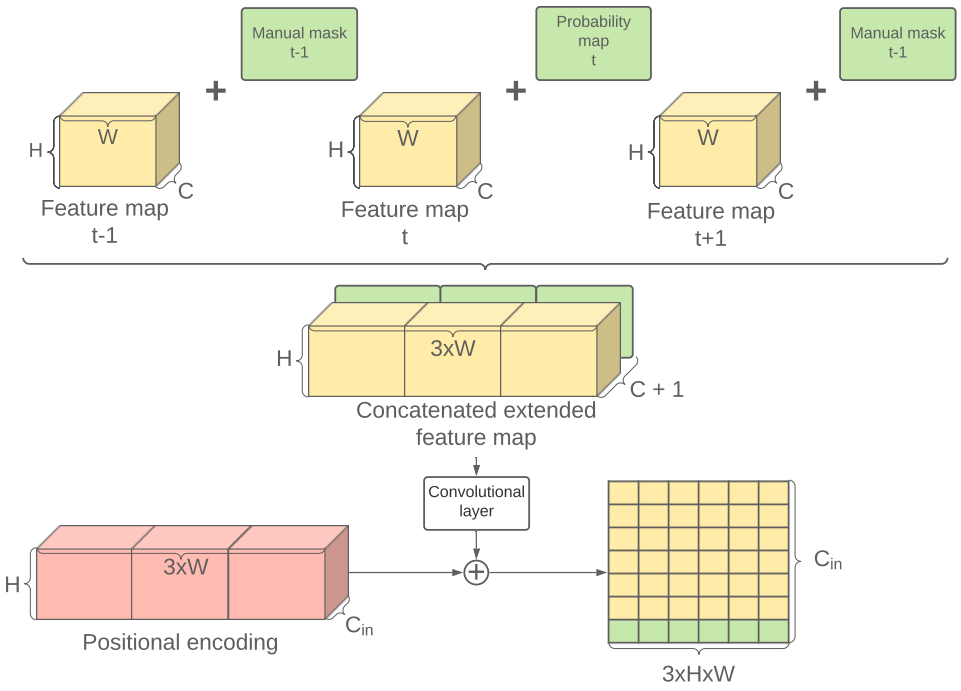


Figure 3.2: An illustration of the transformation of the feature maps before they are fed into the Transformer. The feature maps are extended with an extra channel before concatenated to one big feature map. The concatenated feature map is fed to a convolutional layer, which changes the number of channels to C_{in}. Positional encoding is further added to the total feature map, which is finally flattened in the spatial and temporal dimensions.

After extending the feature maps, the three resulting tensors are concatenated into one and then passed through a convolutional layer. This convolutional layer makes it possible to change the number of channels in the input to the Transformer, and the number of channels after this convolutional layer is referred to as C_{in}. After this layer, positional encoding is added to the tensor before it is flattened in the spatial and temporal dimensions and then used as an input to a Transformer encoder.

The Transformer encoder produces new embeddings of the input, which are fed to the decoder. The decoder is also provided with the input queries, which are generated from the segmentation mask of frame t created by DeepLabv3 [9]. From this mask, an initial key point representation of the boundary for the middle frame is generated by using Suzuki's algorithm [33] with no approximation, which is implemented in OpenCV [34]. The motivation for not using any approximation when extracting the key points is that this should give the Transformer as many points as possible to create a detailed boundary. The key points are used to sample the corresponding feature vectors from the extended feature map for the middle frame. This tensor is then passed through a convolutional layer to adjust the number of

channels to C_{in}. To these final feature vectors, positional encoding is added before they are fed into the decoder as the input queries.

The Transformer outputs embedded key point offsets, which are fed through a fully connected linear layer, transforming them into predicted offsets of the initial key points. These offsets describe how the boundary of the mask in the current frame, generated by DeepLabv3, should be moved to improve the accuracy of the segmentation. From these offsets, the final key points and the boundary polygon are acquired, from which the mask of the instance in the current frame is produced. When this procedure is done for all of the instances in the middle frame, the final segmentation for this image is assembled from all masks.

3.2 Data

A dataset is needed for training and evaluation of the developed network. Requirements for the dataset are that the images should be in video sequences and have instance segmentation ground truth labels, i.e. a class and instance classification for each pixel in every image. Favourable, the video sequences have to be collected road data, i.e. from a camera located at the front of a car in traffic. The network focuses on the class *car*, meaning that this class has to be well represented in the images.

The dataset chosen is KITTI MOTS [36], which consists of 21 online available sequences with instance segmentation masks and the images have the dimensions 1242x375. The object classes represented in this dataset are *car* and *pedestrian*, along with class *background*. The video sequences are captured at 10 Hz, i.e. 10 frames per second. How this data is preprocessing, postprocessed and augmented are described further in Section 3.2.1, 3.2.2 and 3.2.3, respectively.

3.2.1 Data preprocessing

The dataset KITTI MOTS [36] is preprocessed to fulfil the specified requirements. Since the network is trained to segment cars, all pixels classified as *pedestrian* are changed to *background*. Additionally, the developed network structure operates at one instance separately, meaning that each frame needs to be cropped around all the instances. This is achieved by finding a bounding box around each instance and then cropping each frame around the bounding box. The bounding boxes are computed using the segmentation masks and an offline algorithm implemented in pycocotools [37]. The bounding box size is increased by 10% in both dimensions to include more information about the surroundings of the instance.

Some modifications to the ground truth labels are made to make the data usable for training the network. The ground truth segmentation masks are cropped in the same way as the images since they are used together. Also, the Transformer is trained for a loss that penalises the deviation between the predicted boundary and the ground truth boundary. Therefore, it is necessary to create the ground truth

boundary for each instance. This is obtained by applying Suzuki's algorithm [33] implemented in OpenCV [34] on the cropped instance masks. The implementation of all the data preprocessing is done by leveraging the code implemented in MOTS tools [6] and also pycocotools [37]. An example sequence of three images with the ground truth annotations after the data preprocessing is seen in figure 3.3.

(a) t-1 (b) t (c) t+1

Figure 3.3: Three subsequent cropped ground truth images from the KITTI MOTS dataset [36] with the corresponding masks (blue) and boundaries (green). This is an example with strong similarities between the images, where the main difference is the image size, originating from the different distances the instance is captured from.

The data is split into training, validation and test data by choosing different frame sequences for each dataset. More specifically, 13 sequences are chosen for the training dataset, four for the validation dataset and four for the test dataset. This results in 21089 training images, 2798 validation images and 2973 test images after all the frames have been cropped around the instances. The training images have 62.5 % of the pixels labelled as the ground truth class *car*. It should be noticed that all of these images cannot be used to train the whole network since the Transformer needs to be trained on three images in sequence at the time. This means that all the cropped images belonging to the first or last frame in each sequence are not used in the Transformer, and hence the number of training, validation and test images are 20127, 2706 and 2869, respectively.

Some additional transformations of the cropped images are made before they are fed into the network structure. The cropped images have different sizes since the instances appearing in each frame are of different sizes. This is illustrated table 3.1, where the cropped images in each of the datasets are divided into different intervals based on pixel area. The input to the network needs to be of the same size and consequently the masks and images have to be resized into a fixed size before being fed into the network. All the images are also normalised, first to a scale between zero and one and, then, by using the mean $[0.485, 0.456, 0.406]$ and standard deviation $[0.229, 0.224, 0.225]$. This normalisation is performed since the pre-trained DeepLabv3 model [9] acquired from PyTorch [35] assumes images normalised in this matter.

Table 3.1: The distribution of image sizes in terms of pixel area for the KITTI MOTS dataset [36] after the data preprocessing.

Datatype	$[0^2, 64^2)$	$[64^2, 128^2)$	$[128^2, 256^2)$	$[256^2, 512^2)$	$[512^2, \infty^2)$
Training	56.11 %	25.22 %	15.17 %	3.49 %	0 %
Validation	64.97 %	30.74 %	3.47 %	0.82 %	0 %
Test	69.42 %	17.76 %	8.81 %	4.0 %	0 %

3.2.2 Data postprocessing

The desired output from the network is acquired by data postprocessing. The output from the Transformer is key points representing the boundary of the instance. This boundary polygon is used to produce a segmentation mask by utilising a function in OpenCV [34]. Some polygons have to be smoothed after being updated by the Transformer. This is done by applying a one-dimensional Gaussian filter for both the height and the width dimension separately. The kernel size has to be tuned differently depending on the original image size of each cropped image. Empirically it is found that if the original dimension, h or w, is bigger than the resolution used in the Transformer, h_T or w_T, smoothing is applied. Dimensions smaller than the resolution of the Transformer does not need any smoothing. The standard deviation for the Gaussian distribution $\sigma_{h,w}$ is found empirically and is defined as

$$\begin{cases} \sigma_w = 0.8\frac{w}{w_T}, \ if \ w > w_T \\ \sigma_h = 0.8\frac{h}{h_T}, \ if \ h > h_T. \end{cases} \tag{3.1}$$

Even though the performance of the network is evaluated using the cropped images, it can be valuable to inspect the result for the original images visually. Consequently, the masks of each instance from the cropped images are assembled into the original images with 1242x375 dimensions. All pixels that are not predicted to be an instance are labelled as class *background*.

3.2.3 Data augmentation

The developed network structure needs to be trained on sequential data of cars in traffic for the segmentation task. The online public available datasets that fulfil these requirements are limited and are small. This results in that only 21089 images are acquired from the original dataset to train the network. Therefore, data augmentation methods are implemented to increase the training dataset. However, since the project utilises sequential information, only transformations of the data keeping the sequential information between the frames are applicable. Further, to be useful for training, the images should still be reasonable representations of cars. This means that transformations such as flipping the images upside down, rotation and similarly are not good choices. With this in mind, all the images can be mirrored in the training dataset, which would result in that the number of training images is doubled to 42168.

The backbone of the network DeepLabv3 [9] processes each image separately and,

therefore, does not utilise the sequential information. This means that the backbone can be trained on semantic segmentation datasets, making the number of available datasets less limited. Therefore, the Cityscapes [38] dataset can be added to the training dataset aggregated with the mirrored images. In this case, the dataset is processed in the same manner as the original data, and the final number of training images is increased to 73760.

3.3 Training and Evaluation

Due to memory limitations, the backbone is trained separately and then frozen while training the Transformer module. The backbone parameters are trained and updated using the pixel-wise binary Cross-Entropy loss described in Section 2.6.1 together with the Adam optimiser, both implemented in PyTorch [35]. Instead, the Transformer is trained with the Chamfer distance loss L_C described in Section 2.6.2 implemented in PyTorch3D [39], together with the Deviation loss described in 2.6.3. A weight W_{L_D} is used to adjust the impact each of the losses has on the update of the parameters. Therefore, the final loss L used is described as following

$$L = L_C + W_{L_D} L_D. \tag{3.2}$$

The Transformer parameters are updated using the loss L and the Adam optimiser. The backbone and the Transformer are trained until the training loss has converged or the validation loss has not improved in a few epochs. This indicates that the optimum has been found or that the network has started to overfit to the training data. The model with the best validation loss is saved in each case and is used to evaluate the metrics. Both the output from the backbone and the Transformer are evaluated using the Jaccard index and the Contour accuracy, where the latter is implemented using the existing implementation in [11]. In [11] is binary dilation used when calculating the Contour accuracy, with a disc with radius r_{disc} defined as

$$r_{disc} = \left\lceil 0.008\sqrt{h^2 + w^2} \right\rceil. \tag{3.3}$$

Both of the metrics are defined and described in Section 2.8.

4
Results

This chapter describes the experiments determining the network structure and the results from the evaluation of the performance. The backbone was trained separately and then frozen during the training of the Transformer. Hence, the experiments and results are presented in the corresponding order in Section 4.1 respectively Section 4.2. Further, experiments were performed to evaluate the impact of providing the network with sequential information and manual input, which is found in Section 4.3.

4.1 Backbone

To initiate a first guess of the masks, the best way to utilise the backbone was determined by experimenting with the input size, the resizing interpolation method of the masks and data augmentation. These experiments and their results are found in Section 4.1.1. Based on these results, the most promising model was chosen, and its result on the test dataset was used as a benchmark throughout the project. The performance of the chosen model structure is presented in Section 4.1.2, including example images showing some typical segmentations produced by the model.

4.1.1 Determining model structure

The backbone network described in Section 3.1.1 was trained independently from the rest of the network, with several different setups. Some experiments, that are left out from this thesis, were executed to tune the hyperparameters learning rate, batch size and dropout. It was empirically found that a learning rate of 10^{-5} with a decay of 0.5 at every 6th step and a batch size of 20 worked satisfactory well. The training setups for the models are described further in this section. The validation dataset was used to evaluate the models, and the results are summarised in table 4.1.

The first model, B1, was trained with the input size 128x128 of the cropped images. The data used for training was the preprocessed training dataset described in Section 3.2 without any data augmentation. The ground truth masks used in the loss function was resized with bilinear interpolation. This resulted in a Jaccard index of 79.32 % and Contour accuracy of 66.65 %. The training loss and the validation loss had a noteworthy difference. Besides, the validation loss had an unstable behaviour and started to increase in the later epochs. All of this indicated that the model was

Table 4.1: A collection of the model structures and training setups, followed by the Jaccard index J and the Contour accuracy F for the models. The validation dataset was used for the evaluation. Data mirror refers to when the dataset was augmented with mirrored images and Data CS to when it was extended with the images from Cityscapes [38]. The interpolation option Nearest refers to nearest neighbour interpolation. Loss diff. refers to the difference between the validation loss and training loss divided by the validation loss.

Model	Crop size	Data mirror	Data CS	Mask interpolation	J [%]	F [%]	Loss diff. [%]
B1	128x128			Bilinear	79.32	66.65	40.70
B2	128x128	✓		Bilinear	79.78	66.49	39.34
B3	128x128	✓		Nearest	83.12	73.86	41.28
B4	128x128	✓	✓	Nearest	83.34	75.85	19.07
B5	**256x256**	✓	✓	**Nearest**	**84.91**	**78.62**	**29.24**
B6	512x512	✓	✓	Nearest	84.61	77.89	28.83

overfitting the training data.

Next, model B2 was trained to deal with the potential problem of overfitting the training data. One way to prevent a model from overfitting is by train it on more data. Hence, the training dataset was duplicated by mirroring all of the images as described in Section 3.2.3. As seen in table 4.1, the mirrored images did not have any significant impact on the metrics. Nevertheless, the losses did not differ to the same extent. Motivated by this, the mirrored images were kept in the training dataset.

Further, model B3 was trained where the ground truth masks used in the loss function was resized with nearest neighbour interpolation. Using this method, instead of bilinear interpolation, achieved binary ground truth masks for the loss. This resulted in a significant increase of the Jaccard index to 83.12 % and Contour accuracy to 73.86 %.

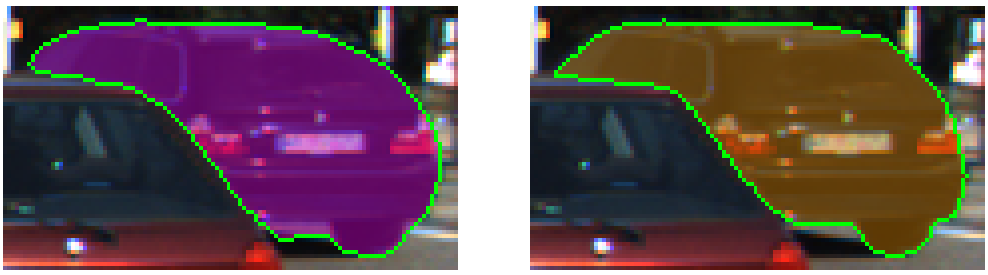
To reduce the overfitting even further, three models, B4, B5 and B6, were trained with the training dataset extended with Cityscapes [38]. The difference between B4, B5 and B6 was that the input size of the cropped images was 128x128, 256x256 respectively 512x512 to evaluate if the resolution affected the performance. From the results, it was seen that adding another dataset to the training data yielded a better performance of the models and reduced the overfitting. It also appeared that the input resolution of 256x256 achieved the best result.

4.1.2 Evaluating model structure

Based on the evaluation of the different model structures, seen in table 4.1, model B5 was chosen as the backbone of the total network. The test dataset was used for

the evaluation of the models, resulting in a Jaccard index of 91.30 % and Contour accuracy of 89.82 %. This was the benchmark method used for the rest of the work.

The masks produced by B5 were analysed visually. Figure 4.1 shows a successful segmentation mask produced by B5, but when compared to the ground truth masks, it is clear that the detail of the contours can be improved. Especially, the detail at the bottom of the car could be a bit tighter, which was a commonly observed behaviour in the dataset. Figure 4.2 demonstrates another successful segmentation of an image using model B5. In this case, the mask has precise contour accuracy, except for the side mirror, which was frequently missed by the network. The mask shown in figure 5.2.2 is another very good case produced by model B5. The segmentation mask is very accurate and captures the side mirror much better than in image 4.2, but has wrongfully predicted a part of the background, noticed in many images in the dataset. Figure 4.4 - 4.5 displays failure images and, clearly, the network had difficulties with occlusion in figure 4.4 and the black edge in figure 4.5, which occurs in some images as a result of the cropping strategy.



(a) (b)

Figure 4.1: A successful segmentation mask and ground truth mask in (a) and (b), respectively.

(a) (b)

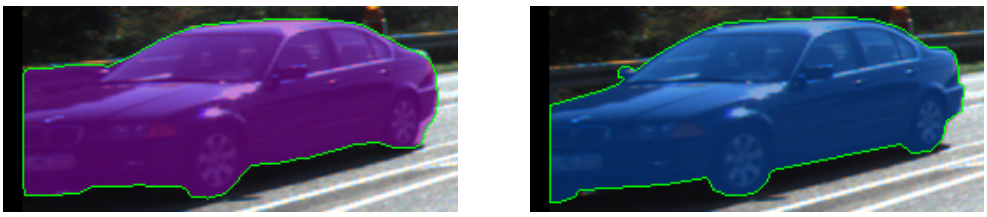
Figure 4.2: A segmentation mask, in which the side mirror was not detected, in (a). The corresponding ground truth mask in (b).

(a) (b)

Figure 4.3: A segmentation mask where part of the *background* was wrongfully predicted as *car* in (a). The corresponding ground truth mask in (b).

(a) (b)

Figure 4.4: A segmentation mask, for which the model struggled with detecting the car due to occlusion, in (a). The corresponding ground truth mask in (b).



(a) (b)

Figure 4.5: A failure segmentation mask and its ground truth mask in (a) and (b), respectively. The model was confused by the additional black pad in the left part of the image.

The pixel-wise confusion matrix for B5, with the test dataset used for the evaluation, is displayed in figure 4.6. The matrix shows that the pixels with ground truth label *background*, 96.06 % were predicted to the correct class and 3.94 % to the incorrect class. The confusion matrix also displays that 97.59 % of the pixels belonging to the ground truth class *car* were correctly predicted, and 2.41 % of them were incorrectly predicted.

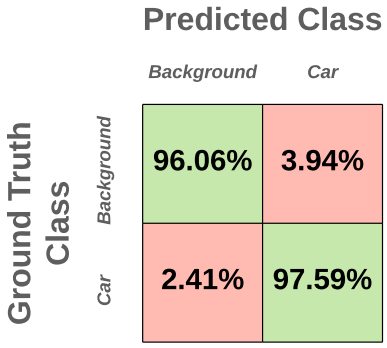


Figure 4.6: The confusion matrix of B5 at the test dataset where the incorrect and correct predictions are shown in percentage in the red and green squares.

4.2 Transformer

This section first determines the most promising model structure in Section 4.2.1 by evaluating different weights on the Deviation loss, Transformer architectures and feature map sizes. The model with the best results based on the validation dataset is further evaluated using the test dataset in Section 4.2.1. Lastly, the model is analysed by visualising the results in different plots.

4.2.1 Determining model structure

Different Transformer models were trained and then evaluated using the validation dataset. All the models had the best DeepLabv3 [9] model as backbone, model B5, and this model is described in table 4.1. The hyperparameters used for training were tuned, and it was found that a learning rate, learning rate decay and learning rate decay step set to 10^{-4}, 0.5 respectively 10 worked satisfactory well. The batch size was limited by the available memory on the computation device used, and was therefore set to the relatively small value 3. A dropout layer with a dropout of 0.05 was used after the multi-head attention layer, in the feed-forward network and after the feed-forward network in both the encoder and the decoder. All the different Transformer setups and their corresponding results is seen in table 4.2 and 4.3 and are explained further below.

The first model T1 was trained using the Chamfer distance loss L_C and the Deviation loss L_D without weighting. The model consisted of one encoder, one decoder and one attention head in each multi-head attention block. Besides the encoder output, the decoder in this model also had 400 input queries as input. The number 400 was set by checking the maximum number of key points on the boundary of the masks produced by the backbone. Both the training and the validation dataset were checked, and this value was rounded up to the nearest hundred. The model T1 resulted in a Jaccard index of 84.56 % and a Contour accuracy of 78.18 %, which was lower than the backbone result, meaning that this Transformer setup made the masks worse.

Table 4.2: The different Transformer model structures. The feature map size refers to the spatial resolution of the feature maps used as input to the Transformer, i.e. the size that the feature maps produced by the backbone were resized to. The Transformer architecture is described by the number of encoders, decoders and heads, where the latter refers to the number of attention heads used in each attention layer. C_{in} is the number of channels in the input feature maps, and queries is the number of queries used in the input to the decoder. The parameter W_{L_D} refers to the weighting of the Deviation loss.

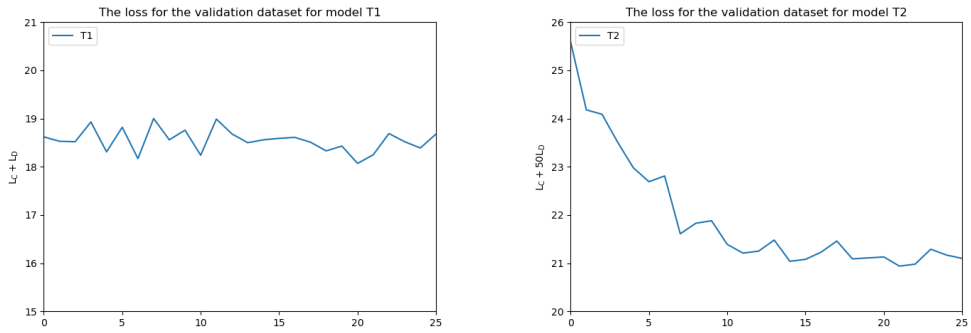
Model	Feature map size	Heads	Encoders	Decoders	C_{in}	Queries	W_{L_D}
T1	64x64	1	1	1	257	400	1
T2	64x64	1	1	1	257	400	50
T3	64x64	1	1	1	257	400	80
T4	64x64	1	1	1	257	400	25
T5	**64x64**	**1**	**1**	**1**	**257**	**400**	**10**
T6	64x64	1	1	1	257	400	5
T7	64x64	2	0	1	256	400	10
T8	128x128	2	0	1	256	1000	40
T9	64x64	4	0	1	256	400	10

Table 4.3: The Jaccard index J and the Contour accuracy F for the Transformer models. The validation dataset was used for the evaluation. The metrics J_s and F_s refer to the Jaccard index and Contour accuracy given after boundary smoothing was applied. The differences in the metrics after the backbone model B5 and the Transformer models are shown in J_{gain}, F_{gain}, $J_{s_{gain}}$ and $F_{s_{gain}}$.

Model	J [%]	J_{gain} [%]	F [%]	F_{gain} [%]	J_s [%]	$J_{s_{gain}}$ [%]	F_s [%]	$F_{s_{gain}}$ [%]
T1	84.56	-0.35	78.18	-0.44	84.36	-0.55	77.02	-1.6
T2	85.75	+0.84	80.35	+1.73	-	-	-	-
T3	84.71	-0.2	77.6	-1.02	84.6	-0.31	76.77	-1.85
T4	86.36	+1.45	81.78	+3.16	86.25	+1.34	81.03	+2.41
T5	**87.1**	**+2.19**	**83.58**	**+4.96**	**86.97**	**+2.06**	**82.7**	**+4.08**
T6	86.49	+1.58	82.49	+3.87	86.33	+1.42	81.5	+2.88
T7	86.89	+1.98	82.98	+4.37	86.75	+1.84	82.03	+3.41
T8	86.91	+2	82.76	+4.14	-	-	-	-
T9	86.88	+1.97	82.83	+4.21	86.72	+1.81	81.85	+3.23

Another Transformer model T2 was trained, with the Deviation loss weight W_{L_D} set to 50. Except for the loss, all other parameters in the Transformer were the same as for model T1. The results acquired for this setup were a Jaccard index of 85.75 % and a Contour accuracy of 80.35 %, meaning that in this case, the Transformer improved the initial masks. When examining the validation loss, it also seemed like the Transformer was able to learn something with this setup, contrary to model T1. For model T2, the loss had a negative gradient almost all the time, while for model

T1, the loss fluctuated around the same value, as seen in figure 4.7.



(a) (b)

Figure 4.7: The validation loss for model T1 in (a) and model T2 in (b). The validation loss in (a) is almost constant, while the loss in (b) clearly decreases.

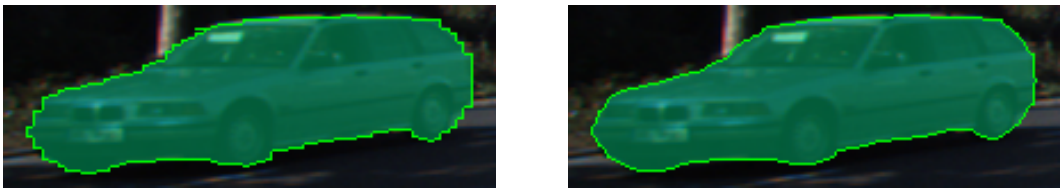
The weighting of the Deviation loss had a large impact on the performance of the Transformer. Hence, model T3-T6 tested different values of the weight. Model T3 was trained with W_{L_D} set to 80, which did not improve the segmentation masks. Since increasing W_{L_D} did not result in better performance, the weight was decreased when training model T4. This model had W_{L_D} set to 25 and gave a Jaccard index of 86.36 % and a Contour accuracy of 81.78 %, which was the best results acquired so far. Therefore, model T5 and T6 were trained with W_{L_D} decreased even further to 10 and 5. Model T5 achieved the best result, with a Jaccard index of 87.1 % and a Contour accuracy of 83.58 %. Hence, 10 was chosen as the best value for the weight.

The rest of the experiments were conducted with W_{L_D} set to 10, which had yielded the best result so far. Two models, T7 and T8, were trained to determine if a larger feature map size improves the performance of the Transformer. Model T8 consisted of one decoder and two attention heads in each multi-head attention block and had feature maps of size 128x128 as input. When using multiple attention heads, the number of channels in the input has to be evenly divisible with the number of heads. Hence, the number of channels C_{in} was set to 256. With the input resolution increased to 128x128, both the height and width are doubled, resulting in a Chamfer distance loss up-scaled with a factor of 4 was needed. The number of queries had to be increased to 1000, following the same methodology as for the smaller resolution. The large size of the feature maps resulted in that an encoder did not fit the memory on the computer device. Therefore, this Transformer did not consist of an encoder. Model T7 had feature maps of size 64x64 and the same architecture as model T8 since this would yield a fair comparison between the models. The results from these models were very similar. T8 had a slightly better Jaccard index but T7 instead had a bit higher Contour accuracy.

Another experiment T9 was executed to evaluate the impact of additional attention heads. This model consisted of one decoder and four attention heads in each multi-head attention block, and the size of the features map was set to 64x64. The

results from this architecture were a Jaccard index of 86.88 % and a Contour accuracy of 82.83 %. This was very similar to model T7, which only differed with two fewer attention heads.

When visually examining the models T1-T9, it was seen that all models apart from T2 and T8 resulted in that large masks had a discontinuous behaviour and looked visually bad. Therefore, smoothing was applied to the boundary of the masks to improve the masks visual appearance. An illustrative example of a mask, before and after the smoothing applied, is displayed in figure 4.8. Even though the masks looked better after the smoothing, the metrics were affected negatively. As seen in table 4.3, the Jaccard index was decreased by about 0.15 percentage points and the Contour accuracy by 1-0.8 percentage points.



(a) (b)

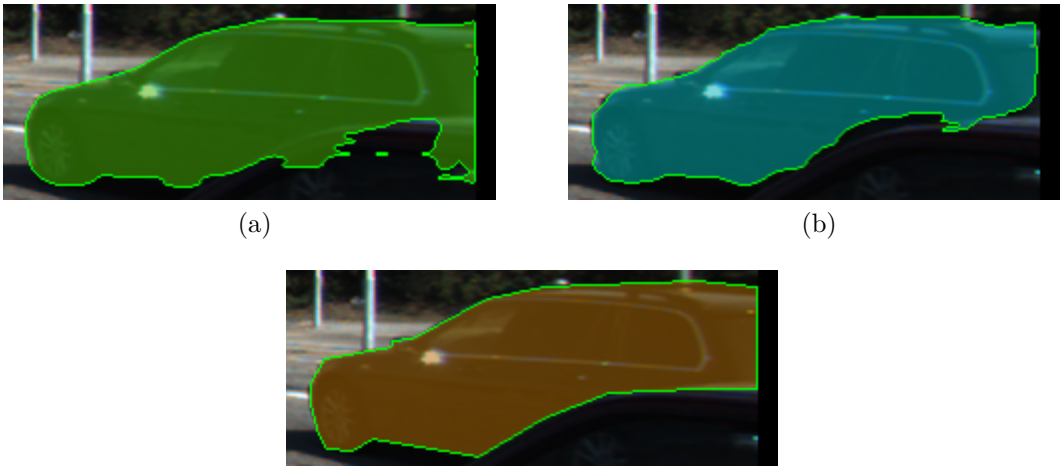
Figure 4.8: A segmentation mask produced by model T5 in (a) and corresponding smoothed mask in (b). The mask in (a) has a discontinuous behaviour, while the mask in (b) has a smoothed boundary and visually looks better.

4.2.2 Evaluating model structure

Model T5 was chosen as the most promising Transformer to use in the total network based on the evaluation of the models T1-T9 using the validation dataset. Therefore, the performance was evaluated using the test dataset resulting in a Jaccard index of 92.65 % and Contour accuracy of 92.78 %, after smoothing applied. This was a gain of 1.35 and 2.96 percentage points respectively compared to the benchmark model B5. The performance of model T5 is analysed further below, and this analysis was mostly done using the Jaccard index since this metric is more conventional. Further, in the previous experiments, the Jaccard index and Contour accuracy had followed the same pattern.

Firstly, some of the masks that been improved by the Transformer were visually analysed. In figure 4.9 - 4.11, some examples of segmentation masks improved by model T5 according to the Jaccard index are displayed. These examples show behaviours frequently occurring over the dataset. In figure 4.9, it is illustrated that T5 has more accurately annotated the boundary towards the other vehicle than the backbone model. Another example of when the Transformer captured the boundary better than the backbone is illustrated in figure 4.10, where especially the boundary at the car roof was corrected. A mask that was quite inaccurate segmented by the backbone, since some parts of another vehicle were classified as the instance, is shown in figure 4.11. In the same figure, it should be noticed how the Transformer

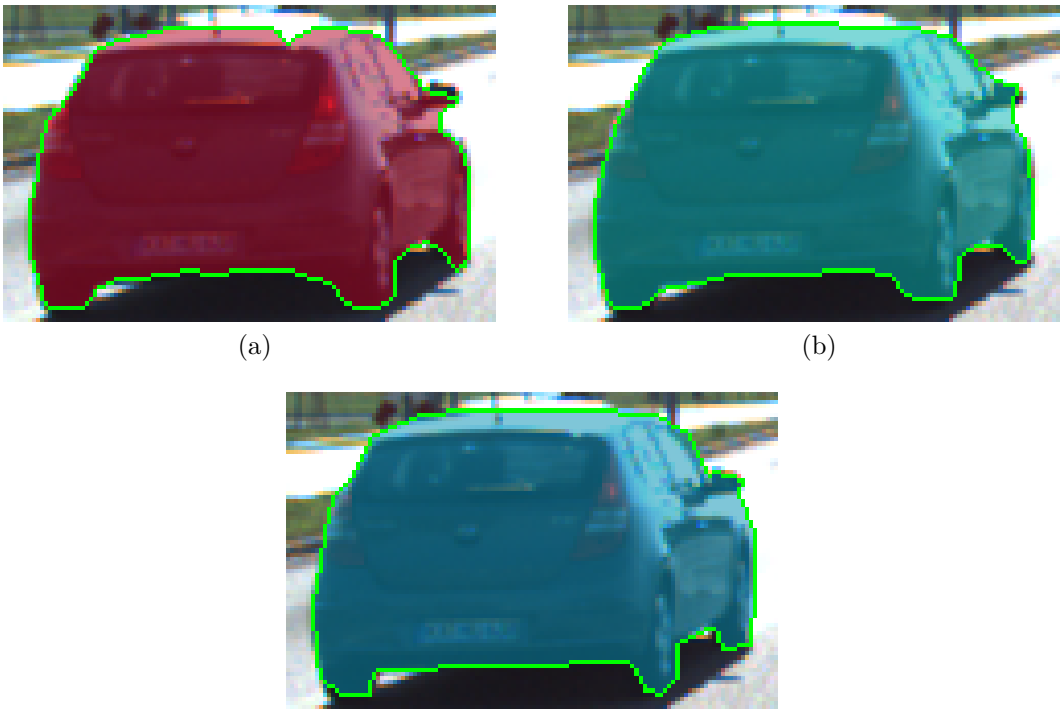
model decreased the number of pixels wrongly annotated in the other vehicle.



(a) (b)

(c)

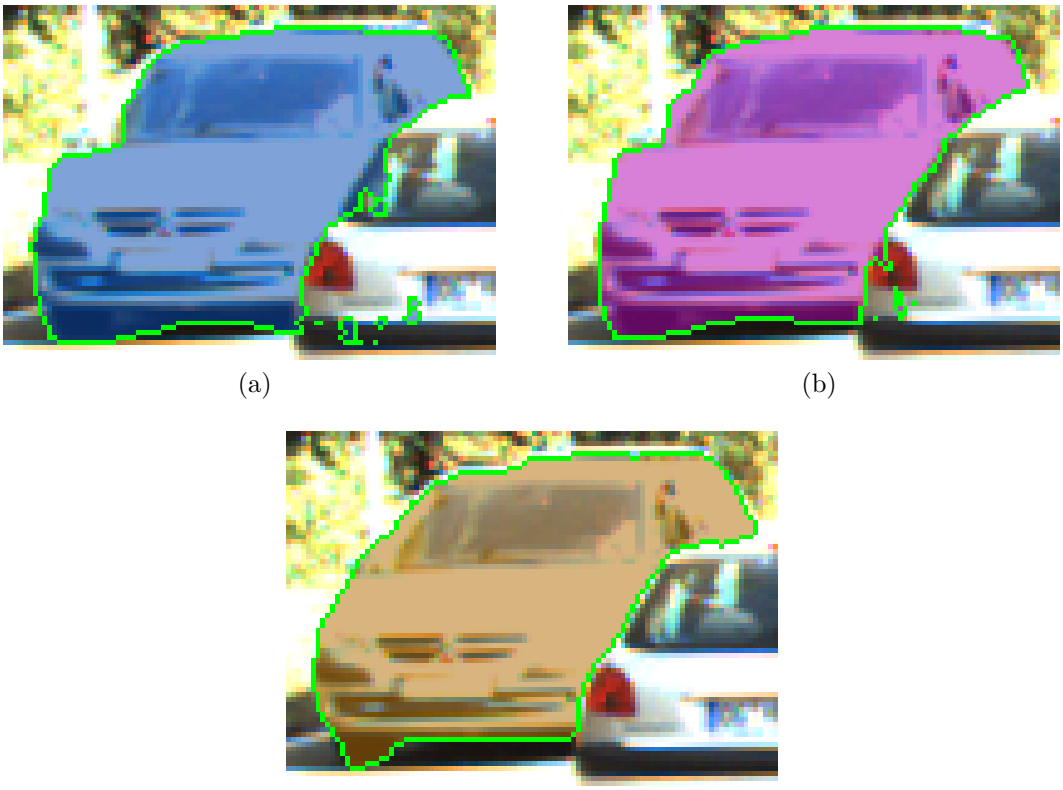
Figure 4.9: A segmentation masks produced by the backbone model B5 in (a), which fails to detect the boundary towards the other car. The corresponding mask improved by T5 in (b) and the ground truth mask in (c).



(a) (b)

(c)

Figure 4.10: A mask annotated by model B5 in (a) and the corresponding more detailed mask created by T5 in (b). The ground truth segmentation mask is illustrated in (c).



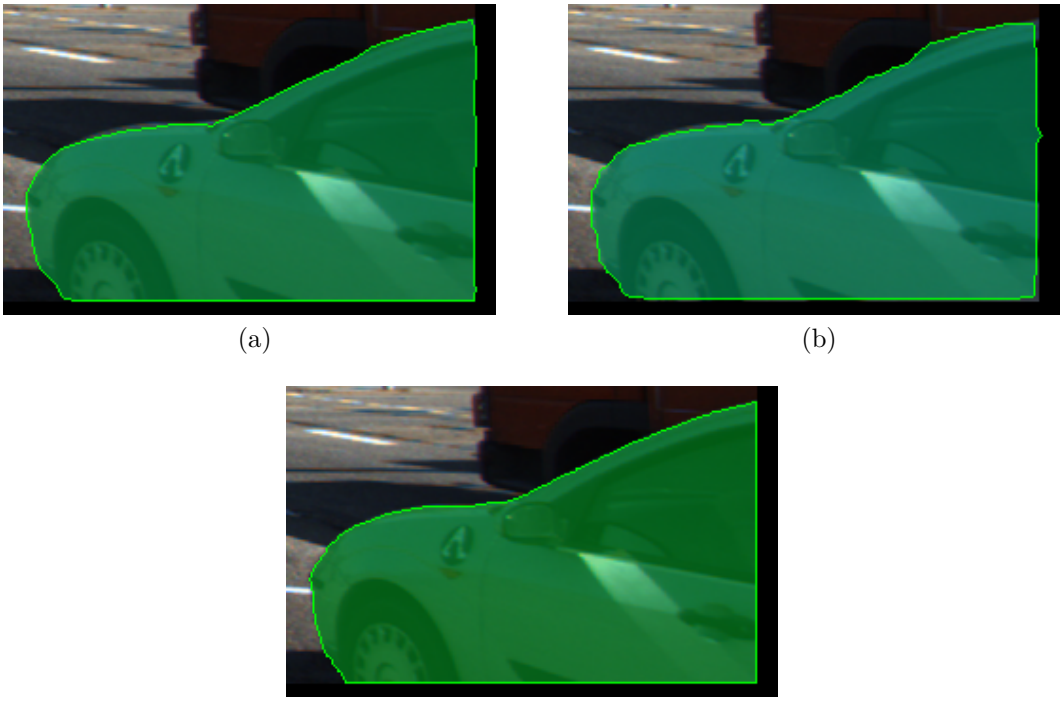
(a) (b)

(c)

Figure 4.11: A segmentation produced by model B5, T5 and ground truth in (a), (b) and (c). Clearly, model T5 reduced the number of pixels in the other car classified as the instance.

The Transformer model T5 did not improve the segmentation masks according to the Jaccard index in all cases. For some masks, it actually decreased the value of this metric. Some examples of frequently occurring behaviours are shown in figure 4.12 - 4.14. Figure 4.12 illustrates a mask that was accurately segmented by the backbone model, while the Transformer decreased the detail of the boundary. An annotation that is not of high quality after the backbone, since the road sign was not detected, is illustrated in figure 4.13. Here, the Transformer decreased the Jaccard index, as it fails to segment this boundary in detail. Still, it found the road sign better than the backbone did. Lastly, figure 4.14 illustrates an example where neither the backbone nor the Transformer detected the rear wheel.

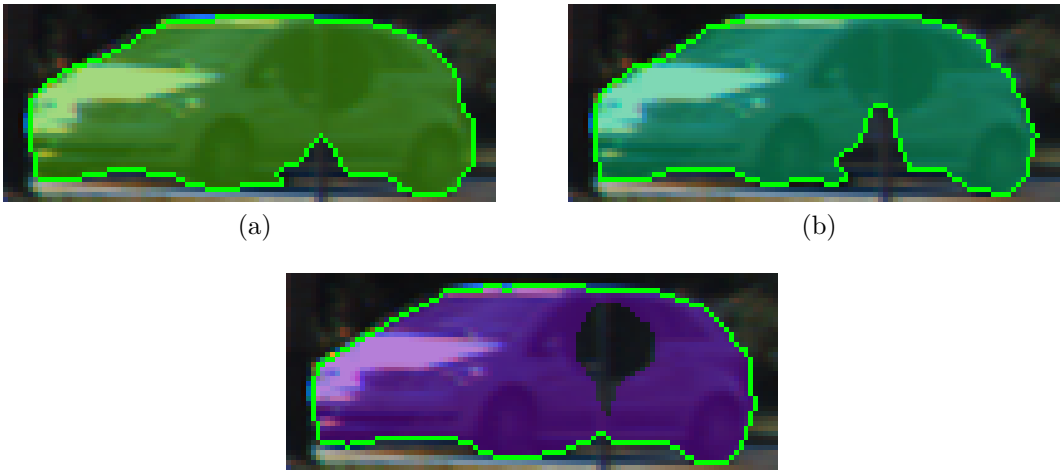
In figure 4.13, it should be noted how the ground truth mask is not precise and how the inner contour of the road sign was not detected in the data preprocessing. This can be explained by the manually annotated mask, which has some pixels at the boundary where the road sign is classified as the instance, and consequently, an inner boundary appears. The data preprocessing, which detects the ground truth boundaries, focus on the outer boundary since it is unlikely for an inner boundary to appear since this means that something flies in front of the car, for example, a bird or similarly. Therefore, the unfortunate result is that the inner boundary was missed in this ground truth mask.



(a) (b)

(c)

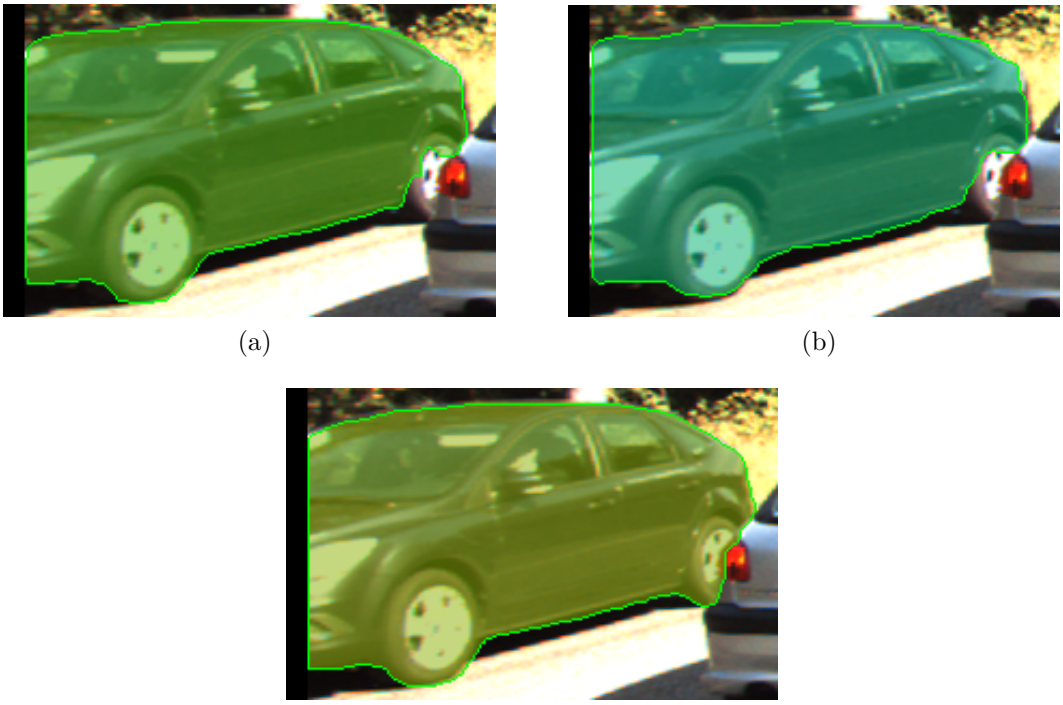
Figure 4.12: An accurate segmentation mask created by B5 in (a) and the worsen masks produced by T5 in (b) together with the ground truth mask in (c).



(a) (b)

(c)

Figure 4.13: Segmentation masks by model B5 in (a), T5 in (b) and ground truth in (c). Clearly, none of the models detected the road sign accurately. It should be noted that even though T5 reduced the Jaccard index, the road sign was better segmented than with model B5.



(a) (b)

(c)

Figure 4.14: A segmentation mask where both the backbone model B5, seen in (a), and the transformer model T5, seen in (b), failed to detect the rear wheel. The corresponding ground truth mask is shown in (c).

After visually analysing the masks, it seemed like many of the images worsened by the Transformer had a high quality after the backbone model. To further investigate this, the Jaccard indexes after the backbone were plotted in a histogram for the masks that the Transformer model decreased the quality of according to this metric. Further, the Jaccard indexes for all the masks were plotted to determine if this distribution differed from the distribution of the Jaccard index for the masks worsened by the Transformer. The histogram is seen in figure 4.15 and clearly, there is a trend that a lot of the images worsened by the Transformer had a very high Jaccard index after the backbone.

To analyse the performance of model T5 further and determine if the Transformer better predicted the class *background*, *car* or neither, a pixel-wise confusion matrix was calculated. This matrix is illustrated in figure 4.16 for the test dataset. The ability to correctly predict certain classes was not significantly changed compared to the backbone.

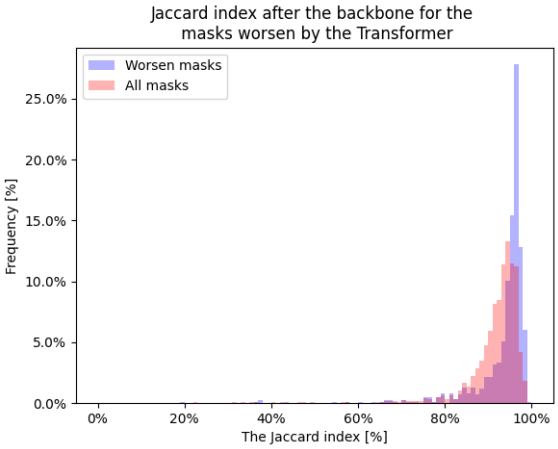


Figure 4.15: The Jaccard index after the backbone for the masks worsened by the Transformer (blue) together with the Jaccard index for all the masks after the backbone (pink). The overlap between the distributions is illustrated in purple. Clearly, a lot of the masks with high Jaccard index get worsen by the Transformer.

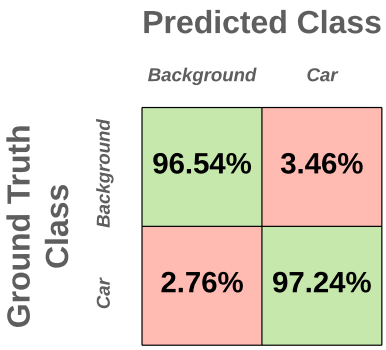
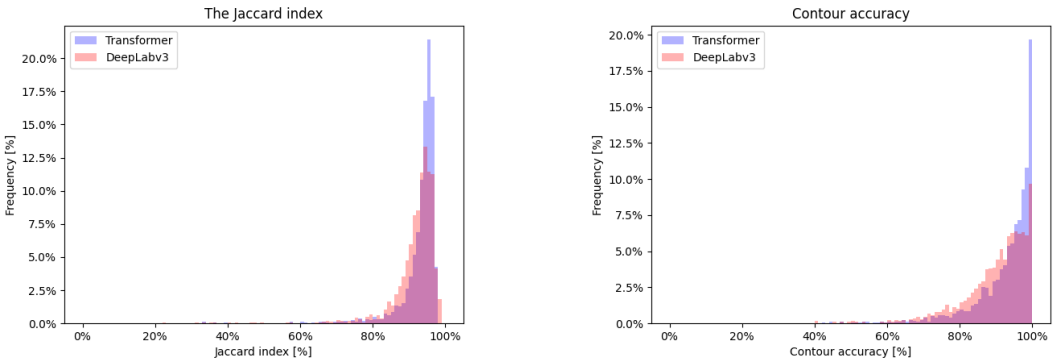


Figure 4.16: The confusion matrix of T5 for the test dataset, where the green squares represent the correctly predicted pixels in percentage and the red squares show the wrongly predicted pixels.

The Jaccard index and Contour accuracy computed are mean values over the test dataset, which does not display the spread in performance. Therefore, two histograms over the metrics were computed to evaluate the distribution for the performance of the backbone model B5 and the Transformer model T5. In figure 4.17, the two histograms are plotted and illustrate the spread in the performance between the two models. Clearly, T5 pushes the curve to the right-hand side, which is desirable. Still, there are some images with Jaccard index 98-99 % after the backbone that were worsened by T5.



(a) (b)

Figure 4.17: The Jaccard index in (a) and the Contour accuracy in (b) for the backbone model B5 and the Transformer model T5, evaluated using the test dataset. The Jaccrad index for the Transformer and backbone model B5 are displayed in blue and pink, while the overlap between the distributions is shown in purple. The Transformer clearly skewed the curve to the right, which is highly desirable.

A histogram was plotted, seen in figure 4.18, to analyse if there is any trend in how much the Transformer model T5 changed the masks. The histogram shows the difference in the Jaccard index in the masks produced by the backbone and the masks adjusted by T5. Clearly, about one-third of the masks were worsened by T5, but the main part of the masks was improved. It is of interest to distinguish if some objects were easier or harder for the network to produce high-quality masks for. Since the total network adjusts images at various original sizes, two scatter plots were plotted to see if image size is a factor for the performance of the Transformer. These plots are shown in figure 4.19, where the image sizes for images that were worsened and improved are shown in separate plots. No distinct trend was identified in the histograms. Still, some more masks of the bigger sizes were worsened than improved and smaller masks were improved slightly more frequently than worsened.

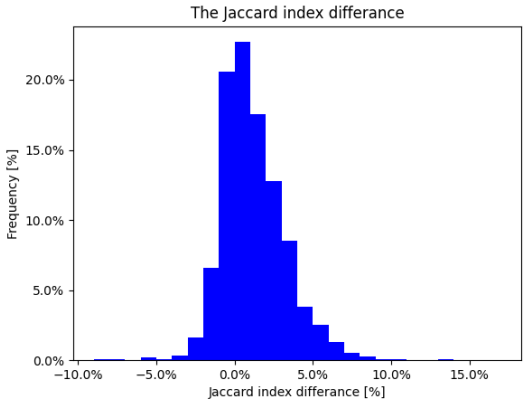
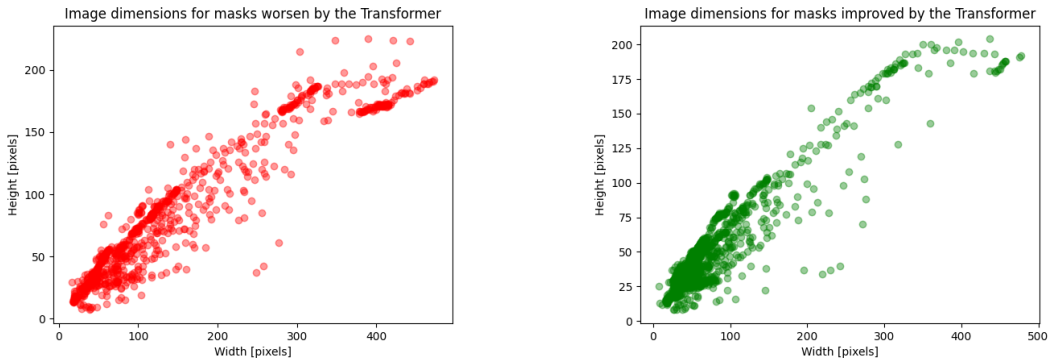


Figure 4.18: Histogram showing the distribution of the difference in Jaccard index between the masks produced by the backbone and the masks adjusted by the Transformer model T5. More masks were improved than worsened by T5.



(a) (b)

Figure 4.19: Scatter plots illustrating the image sizes for images that were worsened (a) and improved (b). There is no distinct trend in which image sizes that were improved or worsened. The number of dots plotted in (a) is about two times the number in (b) since more masks were improved than worsened.

4.3 Sequential information and manual input

Additional experiments were performed to evaluate if sequential information actually helps to improve the segmentations and how important the quality of the information in the extra channel of the extended feature maps are. The experiments were performed with the same setup as for the Transformer model T5, i.e. one encoder, one decoder, one attention-head, resolution 64x64 of the input feature maps, 257 channels, 400 queries and a weighting of the Deviation loss of 10. Also, the hyperparameters used when training were the same as the ones used to train T5, i.e. learning rate, learning rate decay and learning rate decay step set to 10^{-4}, 0.5 and 10, respectively. The total network still consisted of the backbone model B5, while the input to the Transformer model was adjusted in the experiments, which is described further down in this section. The Transformer inputs are gathered in table 4.4, and the results from the evaluation of the models using the validation dataset are found in table 4.5.

First, a model S1 was trained where the information in the extra channel for frame $t-1$ and $t+1$ were exchanged, from the ground truth masks to the probability maps produced by the backbone model B5. This resulted in no significant change in the metrics compared to after the backbone, except the Contour accuracy for the smoothed masks, which decreased.

Another model S2, where the extra channel for $t-1$ was the manually produced ground truth mask, and the mask for $t+1$ was the probability map from the backbone model B5, was trained. With this input, the model learned to improve the masks compared to the backbone, but not to the same extent as model T5, which also had a ground truth mask for $t+1$.

Table 4.4: The different Transformer input setups used to evaluate the impact of sequential information and the information in the extra channel. The option manual refers to the manually produced ground truth mask, while backbone refers to the probability map produced by the backbone. C_{in} refers to the number of channels in the input to the encoder. The Transformer model setups were the ones used to train model T5 and is found in table 4.2.

Model	Input size	Input	Extra channel $t-1$	Extra channel t	Extra channel $t+1$	Nr. frames	C_{in}
T5	64x64	Features	Manual	Backbone	Manual	3	257
S1	64x64	Features	Backbone	Backbone	Backbone	3	257
S2	64x64	Features	Manual	Backbone	Backbone	3	257
S3	64x64	Features	Manual	Backbone	-	2	257
S4	64x64	Features	-	Backbone	-	1	257
S5	64x64	Features	Manual	0.5	Manual	3	257
S6	64x64	Features	-	-	-	3	256
S7	64x64	Features	-	-	-	1	256
S8	64x64	Image	Manual	Backbone	Manual	3	6

Table 4.5: The Jaccard index J and the Contour accuracy F for the models that evaluated the impact of sequential information and the information in the extra channel, evaluated using the validation dataset. The metrics J_s and F_s refers to the Jaccard index and Contour accuracy given after boundary smoothing applied. The difference in the metrics after the backbone and the Transformer models are shown in J_{gain}, F_{gain}, $J_{s_{gain}}$ and $F_{s_{gain}}$.

Model	J [%]	J_{gain} [%]	F [%]	F_{gain} [%]	J_s [%]	$J_{s_{gain}}$ [%]	F_s [%]	$F_{s_{gain}}$ [%]
T5	**87.1**	**+2.19**	**83.58**	**+4.96**	**86.97**	**+2.06**	**82.7**	**+4.08**
S1	84.79	-0.12	78.47	-0.15	84.64	-0.27	77.54	-1.08
S2	85.51	+0.6	79.86	+1.24	85.33	+0.42	78.77	+0.15
S3	86.18	+1.27	81.04	+2.42	86.03	+1.12	80.06	+1.44
S4	84.71	-0.2	78.01	-0.61	84.56	-0.35	76.86	-1.76
S5	86.91	+2	82.92	+4.3	86.76	+1.85	81.94	+3.32
S6	84.75	-0.16	78.48	-0.14	84.61	-0.3	77.63	-0.99
S7	84.84	-0.07	78.72	+0.01	84.69	-0.22	77.7	-0.92
S8	84.62	-0.29	78.15	-0.47	84.4	-0.51	76.96	-1.66

A model S3 was trained with only two input frames, $t-1$ and t, where the prior had the manual mask and the latter the probability map from the backbone. This model adjusted the boundaries in a good direction but not as good as model T5, which had three sequential frames and two manual masks provided. Still, this model had a better performance than S2, which processed three sequential frames with only one manual mask. Further, model S4 was trained without any sequential input, meaning

that the input to the encoder consisted of the feature map for frame t extended with the probability map from the backbone. This model was not able to improve the boundaries from the backbone.

The quality of the information in the extra channel added to the feature maps seemed to have a large impact on the ability to improve the boundaries. Consequently, some experiments were made to conclude if using any probability maps from the backbone should be avoided in the network. Since each feature vector in the input to the Transformer needs to have the same number of channels, the probability map from the backbone in frame t had to be replaced with another tensor if frame $t-1$ and $t+1$ had manual masks. The value zero corresponds to the pixel belonging to the class *background* with 100 % certainty and the value one to the class *car* with 100 % certainty. Hence, another model S5 was trained with all pixels in the extra channel in frame t set to 0.5, since this gave the network information that the mask was uncertain everywhere. This resulted in a model that performed almost as well as T5.

The impact of the masks and the effect of the feature maps produced by the backbone were evaluated even further by training the models S6 and S7, which got the input of three respectively one frame without any extra channel added to the feature maps. These experiments gave similar results, where none of the metrics, except the none smoothed Contour accuracy for S7, were improved compared to the backbone. the none smoothed Contour accuracy for S7 resulted in a small improvement in comparison to the backbone. However, this should not be seen as an actual improvement of the masks since only the smoothed masks have a good visual appearance.

Lastly, a model S8 was trained where the input to the Transformer was three consecutive RGB images extended with the extra channel. The manually produced ground truth masks were used for the frames $t-1$ and $t+1$, while the backbone probability map was used for the frame t. The idea with this test was to evaluate the impact of the feature maps produced by the backbone, with the expectation that the manual masks might have a larger impact if they occupy 1 out of 4 channels instead of 1 out of 257. Also, this experiment was an indication of if the feature maps produced by the backbone model were useful for the Transformer. The positional encoding used during the project included information of the two spatial dimensions and the temporal dimension. It was implemented such that the encoding for each dimension needed at least two channels, i.e. one for the sine wave and one for the cosine wave. Due to this, the number of input channels C_{in} was set to 6. It was empirically found that this setup converged better with a learning rate of 10^{-2} which, therefore, was used in the experiment. The resulting model did not improve the Jaccard index or the Contour accuracy compared to the backbone.

5
Discussion

In this chapter, the chosen methodology and the results are discussed. Section 5.1 analyses the choice of network structure, metrics and dataset. The results from the backbone are treated in Section 5.2, and the Transformer results are discussed in Section 5.3. The experiments considering sequential information and manual input are analysed in Section 5.4. Lastly, ethical and sustainability aspects are taken into account in Section 5.5.

5.1 Methodology

The developed network consisted of a backbone and a Transformer encoder-decoder architecture. DeepLabv3 [9] was chosen as the backbone model since this semantic segmentation network had shown promising results and also was developed to achieve detailed segmentation by utilising atrous convolution. Also, this deep convolutional neural network was available online and hence, could easily be implemented in the project. A Transformer was chosen as the last part of the network as this type of network utilises attention, which feels intuitive to use since the aim of the project was to reason about sequential information. Connections between the frames in the input are needed to utilise this sequential information, which attention seemed suitable for. Also, Transformers had been utilised for segmentation in previous works, such as VisTR [7].

Even though good results were received in the project, by utilising the Transformer, some limitations were introduced by this choice. The Transformer has very high computational complexity due to the attention mechanism, and employing this for images in sequence made the memory on the computer device a limiting factor. Due to this limitation, parameters such as batch size and image size were not possible to experiment with, to any large extent. Though, it is possible to reduce the computational complexity of the attention mechanism if it is replaced with a sparse attention mechanism. This was also taken into account when the Transformer architecture was chosen for the project and thought of as a solution if the shortage of computer memory should have limited the project too much. Fortunately, successful results were acquired without larger images or batch size, and hence a sparse attention mechanism was never prioritised to be implemented.

The project was limited by the available sequential datasets, with ground truth instance segmentation masks, which had to be used to train the Transformer. Fur-

ther, the project only considered the classes *car* and *background*, which made the available annotated images even fewer. If the Transformer would be pre-trained on other classes and then trained on the two chosen classes, it is possible that even better results would have been received. This was not considered since the similarities between cars and other common classes, such as different animals or humans, are not obvious. Therefore, it was assumed that the similarities are too small to achieve a positive impact on the performance of the Transformer by using any of these classes in the training dataset.

The metrics chosen for the evaluation of the model were the Jaccard index and the Contour accuracy, but the latter of the two was less considered. The Jaccard index takes the whole segmentation mask into account, of which the boundary pixels only is a small portion. Therefore, it was thought that an improvement of the boundary accuracy would not result in a significant change of the Jaccard index, leading to the evaluation of the Contour accuracy. Throughout the experiments, the metrics almost exclusively followed the same pattern, i.e. if one of them was increased was the other one also increased. This makes a compelling argument that both of these metrics might not have been needed to evaluate the network.

There was a significant difference in the metrics for the backbone at the test dataset compared to the validation dataset. This is probably due to that one or more of the sequences in the test dataset are similar to some of the sequences in the training dataset. Further, the Transformer improved the metrics for the validation dataset more than for the test dataset, which could be due to the noticed trend that the Transformer more frequently worsens masks of already high quality, which there were more of in the test dataset.

5.2 Backbone

The different model structures tested are discussed in Section 5.2.1, more specifically, different interpolation methods, image size, and data augmentation methods are analysed. Section 5.2.2 discusses the performance of the model that yielded the best results on the validation dataset.

5.2.1 Determining model structure

From the results, it was concluded that changing the resizing method of the masks from bilinear to nearest neighbour had a positive effect for the performance of the network. The reason for this is probably the binary nature of the masks, which is not kept when the bilinear method is used, illustrated in 2.7. The nearest neighbour interpolation retains the binary form of the masks, which probably, is easier for the network to learn from given the results. One reason for this could be that the Cross-Entropy loss is used to train the network. This loss expects the ground truth mask to contain the class that each pixel should be classified as, which means that the ground truth pixel values should be zero or one in this project since only the classes *car* and *background* exists. When the masks are bilinearly interpolated,

other values are introduced, meaning that the Cross-Entropy loss will regard them as separate classes. Thus, the loss is optimised for other classes than solely 0 and 1, which is a possible reason that it is more difficult for the network to learn the correct segmentation. Another thing worth noticing is that the boundary values of the ground truth masks are changed from binary values to float values when bilinear interpolation is used. In this project, the boundary of the mask is of the essence since this is where the network usually struggles to be accurate, meaning that this effect especially is undesirable in this project.

The results from models B4, B5 and B6 showed that an input crop size of 256x256 was the better choice of the tested sizes. The reason could be that in the validation dataset, 4.29 % of the images have a pixel-area bigger than 128x128, only 0.82 % bigger than 256x256 and none bigger than 512x512. All images bigger than the input resolution need to be downsampled, while images smaller than the input resolution need upsampling. When an image is downsampled, information is lost which is not wanted. At the same time, upsampling does not necessarily produce any new valuable information. On the contrary, the information is spread over a larger spatial area when upsampling. Thus, a larger receptive field is needed to achieve the same contextual information. This could be the reason that the largest image size did not result in the best performance. Instead, the optimal input size can be optimised around a value where as little information as possible is lost by downsampling and at the same time upsample as little as possible, which empirically was found to be around 256x256 for this dataset.

The original dataset was augmented by mirroring all the images in the training dataset. This was done to hopefully reduce the overfitting to the training data, which was identified as an issue by comparing the loss in the training dataset to the loss in the validation dataset. It is commonly known that increasing the training dataset usually decreases the overfitting, which it did a little bit in this case. For model B1, the difference between the validation and training loss was 40.70 % of the validation loss, while for the DeepLabv3 model B2, the same difference was 39.34 %. This means that the augmentation of the dataset reduced the overfitting a bit, but not as well as intended. Further, the performance of the network did not improve by adding the mirrored images since the DeepLabv3 model B1 almost had the same Jaccard index as model B2 but a lower Contour accuracy than model B2. The reason for this and that the overfitting only was reduced a bit is probably that the mirrored images are too similar to the original images for the network to learn enough new things to reduce the overfitting satisfactorily and improve the overall performance of the network.

The Cityscapes [38] dataset was added to the training data to further decrease the overfitting. This reduced the difference between the training loss and validation loss a lot. For model B4, the difference between the validation and training loss was 19.07 % of the validation loss. This is much lower than 41.28 %, which was the value for model B3, which was trained with the dataset only extended with mirrored images. This means that the additional data reduced the overfitting a lot. The rea-

son this had a much larger impact than the mirrored images is probably that the Cityscapes images contained enough new information for the network parameters to be fitted for more types of images of cars.

5.2.2 Evaluating model structure

A visual analysis of the masks for the best backbone model B5 is of importance since this is helpful for the understanding of the network. From this, it was seen that the overall performance of the backbone was accurate as the main part of the boundaries were detailed segmented. Nevertheless, some frequent unwanted behaviours were noticed, which are discussed in the following three paragraphs.

The backbone often failed to detailed segment the boundary under the car, as in figure 4.1. Possible reasons for this could be that the car bottom is less continuous and also that the car is hard to distinguish from the road shadowed beneath. This reasoning complies with the case of missing the side mirror, as in figure 4.2, since side mirrors deviate from the otherwise continuous boundary. The reasoning about difficulties distinguishing similar colours is also followed up by figure , where a background car is included in the mask. Detecting side mirrors is particularly hard since side mirrors are not present in all images, for instance, if a car is captured from the side. Hence, the network cannot learn that there always should be a bulge in the boundary to include the side mirror. Further, the number of pixels covering a side mirror is small relative to the total amount of pixels of the car. This means that segmenting side mirrors or not scarcely influences the loss.

Another struggle for the backbone was dealing with occlusion, especially from traffic signs and posts, as in figure 4.4. This possibly originates from many causes. For instance, a sign or post of similar colour to the car can be hard to distinguish. Also, if the blocking object is small relative to the car, convolutional neural networks often have trouble detecting the object due to the downsampling needed to understand the contextual information. DeepLabv3 [9] actually deals a bit with this problem by introducing atrous convolution, but still, the task is hard to solve. Additionally, if the detached area of the car is small, it might be hard for the backbone to get enough contextual information to understand that it is a part of the car. Once more, segmenting a small area or not does not impact the loss enough.

Lastly, the black edges present in some images introduced by the cropping strategy confused the backbone several times, as seen in figure 4.5. A hypothesis is that by cutting the car, segmenting it becomes similar to occlusion due to the lack of contextual information. This means that it is not obvious that it would be easier for the backbone to learn without the black edge since this means that the image ends instead, leading to the same problem.

To investigate the performance of the network even further, the confusing matrix in figure 4.6 is analysed. This shows that the backbone model learned to predict pixels that belong to class *car* better than pixels belonging to class *background*. This

could be a result of the class imbalance in the dataset, i.e. that 62.5 % of the pixels belong to class *car*. The imbalance implies that if the network is unsure of which class to pick, the better guess would be class *car*. Further, since the network was trained on more pixels with the ground truth label *car* than *background*, the network was probably more uncertain when predicting *background*-pixels. Altogether, this is probably the reason that the network confused *background*-pixels with the label *car* more often than the other way around and, consequently, had a lower accuracy when predicting these pixels.

5.3 Transformer

In this chapter, firstly, the behaviour of the different Transformer structures and the reason for the models' different performances are discussed in Section 5.3.1. Then the results from the best Transformer architecture T5 are discussed in Section 5.3.2.

5.3.1 Determining model structure

The evaluation of the performance of models T1-T6, seen in tables 4.2 and 4.3, showed that the weighting on the Deviation loss affects the results a lot. The Deviation loss penalises differences between the distance between the predicted key points and the mean distance between the key points in the ground truth polygon. Further, this loss favours that many points move a little bit, over that few points move a lot, which is very desirable to ensure a smooth boundary. If the weight is set to a low value, the loss has almost no impact meaning that the desired properties it forces upon the network are not gained. If this weight is instead set to a too large value, the key points are not allowed to move much due to the restriction of the distances between them by the loss. This probably affected the network such that it could not move the key points as much as needed, which might be the reason why the network seemed so affected by the weight and why the tuning of it was of importance.

By visually examining the models, it was concluded that all masks, except for models T8 and T2, had a discontinuous behaviour and needed smoothing. Also, there seemed like it only affected the masks with large original image size. This undesirable behaviour is probably due to the upsampling, which occurs when the masks, with a larger original size than the one used in the Transformer, are resized to the original size. Also, this theory is validated by model T8, which outputted masks of a larger size than the rest of the models and did not need smoothing. The larger size of the masks means that less upsampling has to occur and, consequently, the undesirable discontinuous behaviour did not take place. The reason model T2 did not need smoothing must be that a weighting of 50 was used on the Deviation loss. This is very unexpected since all the other models using this resolution of feature maps had different behaviour. Why this occurred for this weight is unknown and was not investigated further in the project since the smoothing satisfactory solved the problem for the other models.

If the metrics before smoothing is considered, the size of the feature maps used

as input to the Transformer did not seem to affect the network's performance considerably. This is showed from the results from model T7 and T8, which was very similar, and if they favour some of the models it is the model with the smaller feature map size. But if the fact that model T8 had smoothed boundaries contrary to model T7 is taken into consideration, T8 is the better model of the two. This is because a fair comparison between these models has to be between the smoothed metrics of model T7 and the non-smoothed metric of model T8 since model T8 did not need any improvement of the visual appearance of the segmentations. Consequently, a larger input size is more desirable for the network's performance. Unfortunately, it also increases the computational complexity of the model yielding a longer training time. With this size of the input, a larger network structure did not fit the computer memory. This means that the network structure used in the best model, T5, was not tested because this model consisted of an encoder. Since a larger size of the feature maps seems to have a positive impact on the performance of the network, even better results would probably be acquired if larger feature maps are used together with T5.

Model T5 had one encoder, one decoder and one attention-head, and if the results from this model are compared with model T7, which did not have an encoder but instead had two attention heads and one decoder, can it be concluded that the encoder has a positive impact on the network's performance. The reason is probably that the additional embedding of the feature maps is useful since it takes contextual information into account and makes it easier for the decoder to predict accurate key point offsets. Further, if the results from model T7 is compared with model T9 which had one decoder and four attention heads, can it be concluded that the number of attention heads does not seem to affect the accuracy of the segmentations noticeably as the results were very similar. It is possible that multiple heads could be useful when the encoder is used, but due to limitations on the computer device, this was not possible to test with this feature map size.

5.3.2 Evaluating model structure

By visually analysing the masks produced by the most promising model, T5, it is seen that the Transformer sometimes improved the masks visual appearance. In the histogram showing the difference of the Jaccard index between the backbone mask and the mask updated by the Transformer, figure 4.18, it is also apparent that the majority of the masks were improved. This result complies with the histogram representing the Jaccard index and Contour accuracy, figure 4.17, and with the mean value of the mentioned metrics.

One example of how the masks were improved is that the Transformer reduces some distortions of the boundary, like the one in figure 4.10. Hopefully, the reason for this is that the Transformer uses the manual masks in the adjacent frames to understand how the vehicle looks in the current frame. In some cases, the Transformer was not able to move the boundary enough to be accurate. For instance, this was the case when the backbone mask was very inaccurate. The reason the Transformer cannot move the boundary enough can be a consequence of the Devi-

ation loss used. This loss favours that many pixels move a little bit over that only some pixels move a lot. This is desirable in many cases, but if the backbone mask is flawed, this loss might not be helpful since some pixels should move a large distance.

Sometimes, the Transformer was not beneficial for the masks, as seen in figure 4.12 - 4.14. This was also noted when examining the histogram in figure 4.18, which illustrates the difference between the Jaccard index before and after the Transformer. A lot of the masks worsened by the Transformer had a very high accuracy already after the backbone, which is seen in the histogram in figure 4.15. A reason that high-quality masks were worsened might be that the Transformer troubles outputting that no key points should move since its task is to adjust the boundary. Another explanation might be that the sequential information in the adjacent frames is not valuable when the mask already is of very high quality. If this would be the case, the similarities between the frames are probably smaller than the quality of the mask already achieved, meaning that these frames cannot be used to improve the segmentations. For the larger images, the reason that the quality was decreased might be that the boundary of these images can only be described by as many key points that fit the boundary in the image size used in the Transformer. To describe the boundary of these images in their original size, more key points are probably required. Hence, the quality possible to achieve by using the Transformer is lower than the quality after the backbone for these images. This is a consequence of the data preprocessing, which resulted in different image sizes, meaning that the images had to be resized into a fixed input size.

By analysing the confusion matrices for B5 and T5 in figure 4.6 receptively 4.16, no clear difference is seen. From this, it is concluded that the Transformer is not better than the backbone at predicting any of the classes. The false positives in the masks from the Transformer is probably a consequence of the false positives that the backbone predicted. This is expected since the Transformer corrects the boundaries from the backbone model and thus not make any own classification directly.

When studying the histogram of the Jaccard index and Contour accuracy spread in figure 4.17, it is concluded that the Transformer produced more masks with high quality than the backbone did. Still, some masks with top quality after the backbone was worsened in terms of the Jaccard index by the Transformer. This might be due to a Jaccard index close to 100 % is difficult to achieve, and the quality of the ground truth mask affects the last percentage points a lot. It might be interesting to investigate how much the Jaccard index differs between two manual annotators to set a threshold for when a mask is of ground truth quality. From the mean of the Contour accuracy, it is seen that the Transformer improved the boundaries, which was the intention for the model. It is also seen that a Contour accuracy of 99-100 did not automatically yield a Jaccard index at the same level. This could be due to the binary dilation used to thicken the boundary when computing the Contour accuracy, resulting in that a good mask becomes perfect. The little displacement of the boundary still impacts the Jaccard index since the mask is not adjusted before calculating this metric.

The scatter plots in figure 4.19 illustrate the image sizes of the masks that were worsened and improved, showing that masks of all sizes were adjusted in both directions. Looking closely, a few more large images were worsened than improved. The cause for this is not established, but some hypotheses are evolved. The first theory is that the training dataset consists of many more small images. Hence the network was mainly trained for smaller sizes, leading to better performance for these. But since all of the images were resized to the same resolution, it is not obvious that the original image size matter. Another hypothesis is that large objects are closer to the camera and, hence, more often changes a lot between frames. Since the Transformer appears to utilise information between frames, it seems natural that it is harder to leverage sequential information for objects that change much between frames.

5.4 Sequential information and manual input

By comparing models S1-S8 with each other and T5, some theories about the importance of the manually annotated masks, the sequential information and the feature maps for the network's performance are formed. The models are described in table 4.4, and their results are found in table 4.5. The analysis is described in the paragraphs below.

Model S1, which had three consecutive frames with probability maps produced by the backbone as input, did not improve the backbone masks. Also, the resulting Jaccard index and Contour accuracy were lower than after the backbone. The reason that this model performed much worse than T5 might be the lack of manually annotated masks since this is the only difference between these two models.

When the Transformer instead had one manual mask as input together with two probability maps from the backbone, as in model S2, the backbone masks were improved a little bit but not as much as when using two manually annotated masks in the input. It, therefore, seems like the Transformer uses the information in the extra channel. When the information added in the extra channel is not of high quality, the Transformer is confused.

That the quality of the information in the extra channel is important for the performance of the Transformer is also validated by model S3, which only had two frames as input, where one frame had a manual mask and one a probability map produced by the backbone. This model gave a better performance than model S2, which strengthens the hypothesis that information of lower quality than ground truth only confuses the Transformer. The reason the backbone probability maps in the adjacent frames are not helpful for the Transformer might be that the information needed to improve the mask in the middle frame is not given in the backbone probability maps. If the backbone has difficulties segmenting some part of the car, it is likely that this problem also occurs in the adjacent frames and hence the information of how to annotate this part of the vehicle is not given to the Transformer.

Model S3 improved the backbone masks but not as much as model T5. Since the only difference between the models is that T5 had two manually annotated masks and S3 had one, it seems like the Transformer utilises the information in both the adjacent frames when annotating the middle frame. The more high-quality information given to the Transformer, the better. The reason for this could be the attention mechanism in the Transformer, which makes connections between the input and takes the contextual information into account. More temporal information means more contextual information and, consequently, better connections between the input can be formed by the Transformer. Also, this is showed by the results from model S4, which only had the frame that is to be annotated as input and only lowered the metrics compared with the backbone model. Without temporal information, the Transformer cannot make sufficient connections to yield improvements of the segmentations. Also, this means that the attention over the frame itself does not give any new helpful information compared with the contextual information the backbone found. It is the temporal connections that are made in the Transformer that is of the essence for the performance.

An experiment with two manual masks and the value 0.5 everywhere in the extra channel of the middle was conducted in model S5. The idea behind the value 0.5 was to let the Transformer know that the network was uncertain about this segmentation. The results from this experiment were very similar to model T5, even though T5 was the superior model. This indicates that the information given in the middle frame also affects the performance, but it is not as essential as the information given in the adjacent frames. This might be explained by the fact that the information about the segmentation in the middle frame is given to the network since this boundary is used when forming the input queries.

Three experiments S6-S8 were conducted to conclude the importance of the feature maps produced by the backbone. Model S6 and S7 had no extra channel added to the feature maps and therefore only consisted of the features extracted from the backbone. Model S6 had three consecutive frames as input, and the results from this experiment were almost the same as for model S1, i.e. no improvement of the backbone masks. Instead, only one frame was used as input to model S7, and the results from this model were similar to S6. Once again, the results showed that the extra channel is important to the performance of the network and the temporal information in the feature maps are not helpful for the network without the extra channel. However, the feature maps still have a positive effect on the network's performance. This is shown by model S8, which had the RGB images as input to the network. The convolutional layer before the Transformer transformed the RGB images into a feature map consisting of 6 channels. This is due to the positional encoding, which needed the input to be of at least 6 channels because the spatial and temporal dimensions were encoded using both sine and cosine functions. This model also had three consecutive frames, where two have manual masks and one probability mask produced by the backbone model. The result from model S8 was the worst result of all the models S1-S8, meaning that the features produced by the backbone model still are helpful for the network.

5.5 Ethical and sustainability aspects

The thesis aims to automate the annotation process further, which might lead to a decrease in the need for manual labour and, consequently, fewer employees. This will probably result in some workers losing their jobs and income. The first UN sustainability goal has the objective to end poverty [40], which this thesis can be seen to oppose. It is also possible that this sustainability goal is not affected at all. This can be the case if companies buying the annotated data still are willing to spend the same amount of money on data. Consequently, manual labour will still be needed to the same extent as more data will be bought, and hence no employees will lose their jobs.

Annotated data is utilised to train algorithms used in advanced divining assistance systems and for the development of self-driving vehicles. Hence, the ethical and sustainable aspects of automated cars have to be investigated. Firstly, cars that are automated, or have automated safety systems, have the possibility to be safer than manually driven. Self-driving vehicles also enable passengers to spend time on other things than driving, resulting in transportation that is more efficient and might lead to increased well-being. Altogether this has a positive impact on the UN's third sustainability goal, which has several objectives such as decreasing accidents in traffic and encouraging mental health [41]. The development of self-driving or highly automated cars requires new technology and can, therefore, also be recognised as contributing to the ninth UN sustainability goal since it, among other things, aims to encourage innovation [42].

Safer and more efficient cars could result in a higher usage of cars, which harms the environment if non-renewable energy resources are used. Though it is possible that self-driven cars result in a new way of viewing ownership of a car, and it is possible that multiple persons will own the same vehicle or that society will utilise the cars as public transportation. In any case, this would most likely lead to a decrease in the number of newly bought cars, which will have a positive impact on the environment. The realisation of totally automated vehicles will therefore probably have some impact on the thirteenth UN sustainability goal, which aims to reduce the negative impact on the environment [43].

6

Conclusion

In this chapter, the conclusions drawn from the project are presented. The research questions formed at the beginning of the project are answered in Section 6.1. Suggestions for how to proceed from this project are displayed in Section 6.2.

6.1 Conclusions from results and discussion

The developed network structure performs well and improves many of the segmentation masks compared to the benchmark method, which is the same as the backbone model. The final network structure consists of DeepLabv3 [9] and a Transformer that includes an encoder, a decoder and one attention head in each attention layer. The input to the whole network is images of size 256x256x3. The input to the Transformer encoder is three flattened feature maps, each having the resolution 64x64x257 before being flattened in the spatial and temporal dimensions. The decoder takes the output from the encoder as an input, together with 400 input queries. Unfortunately, the developed network introduces an undesirable discontinuous behaviour in the produced masks, which, on the other hand, is reduced by a smoothing filter.

The network achieved a Jaccard index of 92.65 % and a Contour accuracy of 92.78 % when the test dataset was used for evaluation. This corresponds to an improvement of 1.35 respectively 2.96 percentage points compared with the benchmark model. It is noticed that not all of the masks produced by the backbone model was improved by the Transformer, and some were actually worsened. These masks were mainly the ones of already very high quality from the backbone. It is therefore concluded that the Transformer model is not necessarily useful for segmentations of high quality.

From the experiments considering different types of sequential information, it is concluded that sequential information is beneficial for the developed network structure. It is also seen that the quality of the sequential information is of the essence. The manual masks are the most important part of the input, but they are not helpful without the features extracted from the backbone. The number of manually annotated masks in the input has a large influence on the network. Decreasing the manual masks to one reduces the accuracy of the network, and removing both of the manual masks decreases the accuracy even more.

6.2 Future work

In this project, segmentation has been in focus. The tracking was assumed to be solved by the human annotator, who also provided the network with bounding boxes. Future work could be to extend the network with a prior network that produces bounding boxes automatically and solves the tracking. Additionally, a solution for overlapping masks in the assembled images could be developed. Suggestions for automatic solutions are either a decision algorithm or a refinement network. Another solution could be to manually solve these parts of the boundaries.

The final network structure benefits both from the manual masks in the adjacent frames and the image embeddings in terms of feature maps extracted from the backbone. The effect of the probability map extracted from the backbone model and added to the feature map in the middle frame seems very limited. It could hence be of interest to investigate if the reason for this is that the network has trouble understanding how to use this information since it is not a binary mask. An idea is for this is to extend the feature map with the mask produced by the backbone instead of the probability map.

As mentioned, both the manual masks in the adjacent frames and the feature maps have a positive effect on the network. Hence the Transformer probably could be provided with additional information to perform even better. An idea of such information is optical flow, used by VGCN [8], which could help connecting pixels between frames. Another idea is to extend the feature maps with a binary tensor representing the boundary instead of the masks as in the current network.

The Transformer improved most of the masks, but in some cases, it did not move the boundary enough to achieve an accurate segmentation. Therefore, it could be interesting to see if multiple Transformers could be used, each correcting the masks produced by the backbone a little bit. This can be implemented by training one Transformer until convergence and then pass all the data through this network. The output from the first Transformer can then be used to train another Transformer, and, if successful, this procedure could be repeated until the accuracy no longer improves.

The positional encoding used in the project encodes temporal and spatial information in the input. This positional encoding is added to the input to the encoder, some of the intermediate layers in the Transformer and the input queries. These input queries are feature vectors for the middle frame, extended with the probability map produced by the backbone, sampled at the key points. The input queries do consequently have an order relative to each other, and it might be helpful to also encode this order in the positional encoding.

The results indicate that larger images and an encoder are beneficial for the Transformer. But due to the high computational complexity in the Transformer, a Transformer consisting of an encoder operation on larger images did not fit the memory

on the computer device. Hence experiments could be executed to determine if this setup achieves even better results. For this, a computer with larger memory could be needed. Another suggestion is to reduce the computational complexity originating from the attention mechanism by replacing the full attention with some sparse attention. Then, would hopefully both large images and an encoder-decoder structure fit on the memory. Both of these suggestions would probably make it possible to experiment with multiple encoders, attention heads, and larger batch size.

Bibliography

[1] X. Du et al., "SpineNet: Learning Scale-Permuted Backbone for Recognition and Localization," in *2020 IEEE/CVF Conf. CVPR*, 2020, pp. 11589-11598, doi: 10.1109/CVPR42600.2020.01161.

[2] R. Mohan and A. Valada, "EfficientPS: Efficient Panoptic Segmentation," in *Int. J. Comput. Vis.*, 2021, vol. 129, pp. 1551–1579, doi: 10.1007/s11263-021-01445-z.

[3] G. Ghiasi et al., "Simple Copy-Paste is a Strong Data Augmentation Method for Instance Segmentation," 2020, *arXiv:2012.07177v1*.

[4] L. Yang, Y. Fan and N. Xu, "Video Instance Segmentation," in *2019 IEEE/CVF ICCV*, 2019, pp. 5187-5196, doi: 10.1109/ICCV.2019.00529.

[5] G. Bertasius and L. Torresani, "Classifying, Segmenting, and Tracking Object Instances in Video with Mask Propagation," *2020 IEEE/CVF Conf. CVPR*, 2020, pp. 9736-9745, doi: 10.1109/CVPR42600.2020.00976.

[6] P. Voigtlaender et al., "MOTS: Multi-Object Tracking and Segmentation," in *2019 IEEE/CVF Conf. CVPR*, 2019, pp. 7934-7943, doi: 10.1109/CVPR.2019.00813.

[7] Y. Wang et al., "End-to-End Video Instance Segmentation with Transformers," 2020, *arXiv:2011.14503v2*.

[8] Y. Xu, Y. Wu, N. S. b. Zuraimi, S. Nobuhara and K. Nishino, "Video Region Annotation with Sparse Bounding Boxes," presented at the BMVC 2020, (Virtual), Sep. 7-10, 2020, Paper 638.

[9] L.-C. Chen, Y. Zhu, G. Papandreou, F. Schroff and H. Adam "Encoder-Decoder with Atrous Separable Convolution for Semantic Image Segmentation", in *ECCV 2018*, 2018, pp. 801-818.

[10] P. Jaccard, "THE DISTRIBUTION OF THE FLORA IN THE ALPINE ZONE.1," in *New Phytologist*. vol. 11 (2), pp. 37–50. [Online]. Available: `https://nph.onlinelibrary.wiley.com`, doi:10.1111/j.1469-8137.1912.tb05611.x.

[11] F. Perazzi, J. Pont-Tuset, B. McWilliams, L. Van Gool, M. Gross and A. Sorkine-Hornung, "A Benchmark Dataset and Evaluation Methodology for Video Object Segmentation," in *2016 IEEE Conf. CVPR*, 2016, pp. 724-732, doi: 10.1109/CVPR.2016.85.

[12] K. He, G. Gkioxari, P. Dollár and R. Girshick, "Mask R-CNN," in *2017 IEEE ICCV*, 2017, pp. 2980-2988, doi: 10.1109/ICCV.2017.322.

[13] J. Liang, N. Homayounfar, W. -C. Ma, Y. Xiong, R. Hu and R. Urtasun, "PolyTransform: Deep Polygon Transformer for Instance Segmentation," in *2020 IEEE/CVF Conf. CVPR*, 2020, pp. 9128-9137, doi: 10.1109/CVPR42600.2020.00915.

[14] A. Vaswani et al., "Attention is all you need," in *NIPS'17*, 2017, pp. 6000–6010, doi:10.5555/3295222.3295349.

[15] S. Zheng et al., "Rethinking Semantic Segmentation from a Sequence-to-Sequence Perspective with Transformers," 2020, *arXiv:2012.15840v1*.

[16] N. Carion, F. Massa, G. Synnaeve, N. Usunier, A. Kirillov and S. Zagoruyko, "End-to-End Object Detection with Transformers," in *ECCV 2020*, 2020, pp. 213-229.

[17] H. Ling et al., "Fast Interactive Object Annotation with Curve-GCN," in *2019 IEEE/CVF Conf. CVPR*, 2019, pp. 5252-5261, doi:10.1109/CVPR.2019.00540.

[18] E. Ilg, N. Mayer, T. Saikia, M. Keuper, A. Dosovitskiy and T. Brox, "FlowNet 2.0: Evolution of Optical Flow Estimation with Deep Networks," in *2017 IEEE Conf. CVPR*, 2017, pp. 1647-1655, doi: 10.1109/CVPR.2017.179.

[19] I. Goodfellow, Y. Bengio and A. Courville, " Convolutional Networks," in *Deep Learning*, Cambridge, MA, USA, MIT Press, 2016, ch. 9, pp. 326-366. Accessed: Mars 17, 2021 [Online]. Available: `http://www.deeplearningbook.org`

[20] K. Bai, "A Comprehensive Introduction to Different Types of Convolutions in Deep Learning." towardsdatascience.com. `https://towardsdatascience.com/a-comprehensive-introduction-to-different-types-of-convolutions\-in-deep-learning-669281e58215` (accessed Apr. 23 2021).

[21] K. He, X. Zhang, S. Ren and J. Sun, "Deep Residual Learning for Image Recognition," in *2016 IEEE Conf. CVPR*, 2016, pp. 770-778, doi: 10.1109/CVPR.2016.90.

[22] L. Chen, G. Papandreou, I. Kokkinos, K. Murphy and A. L. Yuille, "DeepLab: Semantic Image Segmentation with Deep Convolutional Nets, Atrous Convolution, and Fully Connected CRFs," in *IEEE TPAMI*, 2018, pp. 834-848, doi: 10.1109/TPAMI.2017.2699184.

[23] L. Svensson. Transformer - Part 6 - Decoder (1): Testing and Training. (nov. 17, 2020). Accessed: Apr. 23, 2021. [Online Video]. Available: `https://www.youtube.com/watch?v=kpqG7O1dbTk`

[24] L. Svensson. Transformers - Part 7 - Decoder (2): Masked Self-attention. (nov. 18, 2020). Accessed: Apr. 23, 2021. [Online Video]. Available: `https://www.youtube.com/watch?v=piT1_k8b9uM`

[25] L. Svensson. Transformers - Part 1 - Self-attention: an Introduction. (oct. 23, 2020). Accessed: May 17, 2021. [Online Video]. Available: `https://www.youtube.com/watch?v=0SmNEp4zTpc`

[26] A. Dosovitskiy et al., "An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale", presented at ICLR 2021, (Virtual), May 3-7, 2021.

[27] M. Zaheer et al., "Big Bird: Transformers for Longer Sequences", in *NeurIPS 2020*, 2020, pp. 17283-17297.

[28] J. Ho, N. Kalchbrenner, D. Weissenborn and Tim Salimans., "Axial Attention in Multidimensional Transformers", 2019, *arXiv:1912.12180v1*.

[29] X. Zhu, W. Su, L. Lu, B. Li, X. Wang and J. Dai, "Deformable DETR: Deformable Transformers for End-to-End Object Detection", presented at ICLR 2021, (Virtual), May 3-7, 2021.

[30] S. Jadon, "A survey of loss functions for semantic segmentation," in *2020 IEEE Conf. CIBCB*, 2020, pp. 1-7, doi: 10.1109/CIBCB48159.2020.9277638.

[31] E.R. Davis, " Basic Image Filtering Operations," in *Computer and Machine Vision: Theory, Algorithms, Practicalities* 4th ed., Waltham, MA, USA, Academic Press, 2012, ch. 3, pp. 40-42. [Online]. Available: `https://www.sciencedirect.com`, doi:10.1016/B978-0-12-386908-1.00003-3.

[32] "Types of Morphological Operations." MathWorks.com. `https://se.mathworks.com/help/images/morphological-dilation-and-erosion.html` (accessed Mar. 19, 2021).

[33] S. Suzuki and K. Abe, "Topological structural analysis of digitized binary images by border following," in *Comput. Vis. Graph. Image Process.*, 1985, pp. 32-46, doi:10.1016/0734-189X(85)90016-7.

[34] G. Bradski, "The OpenCV Library," in *Dr. Dobb's Journal of Software Tools*, 2000, pp. 120-125.

[35] A. Paszke et al.,"PyTorch: An Imperative Style, High-Performance Deep Learning Library," in *NeurIPS 2010*, 2019, pp. 8024-8035.

[36] A. Geiger, P. Lenz and R. Urtasun, "Are we ready for Autonomous Driving? The KITTI Vision Benchmark Suite," in *2012 IEEE Conf. CVPR*, 2012, pp. 3354-3361, doi: 10.1109/CVPR.2012.6248074.

[37] *pycocotools 2.0* (2018), cocodataset.org. Available: `https://github.com/cocodataset/cocoapi`

[38] M. Cordts et al., "The Cityscapes Dataset for Semantic Urban Scene Understanding," in *2016 IEEE Conf. CVPR*, 2016, pp. 3213-3223, doi: 10.1109/CVPR.2016.350.

[39] N. Ravi et al., "Accelerating 3D Deep Learning with PyTorch3D", presented at the SIGGRAPH Asia 2020, (Virtual), Dec. 4-13, 2020, doi:10.1145/3415263.3419160.

[40] "Sustainable Development Goal 1: End poverty in all its forms everywhere." un.org. `https://www.un.org/sustainabledevelopment/poverty/` (accessed Nov. 12, 2020).

[41] "Sustainable Development Goal 3: Ensure healthy lives and promote well-being for all at all ages." un.org. `https://www.un.org/sustainabledevelopment/health/` (accessed Nov. 12, 2020).

[42] "Sustainable Development Goal 9: Build resilient infrastructure, promote sustainable industrialization and foster innovation." un.org. `https://www.un.org/sustainabledevelopment/infrastructure-industrialization/` (accessed Nov. 12, 2020).

[43] "Sustainable Development Goal 13: Take urgent action to combat climate change and its impacts."un.org. `https://www.un.org/sustainabledevelopment/climate-change/` (accessed Nov. 12, 2020).