KU LEUVEN

FACULTEIT INGENIEURSWETENSCHAPPEN

Enhancing Temporal Consistency in Real-Time Semantic Video Segmentation, Using Recurrent Neural Networks.

Johannes Verherstraeten

Thesis submitted for the degree of Master of Science in Engineering: Computer Science, option Artificial Intelligence

Thesis supervisor:
Prof. dr. M.-F. Moens

Assessors:
Dr. ir. B. De Brabandere
Dr. ir. G. C. Talleda

Mentors:
Dr. ir. M. Proesmans
Ir. J. Heylen

Academic year 2018 – 2019

Preface

I am very grateful for the opportunity of working with deep learning and computer vision. Even after a full year, this topic still excites me and keeps me awake at night, thinking about its endless possibilities. Therefore, I would like to thank my mentors Marc Proesmans and Jonas Heylen very much. They allowed a weird computer scientist student to enter the electrical engineering department. They offered him a lot of freedom and computing power, and provided many helpful ideas. Without them, this thesis would only consist of a title page and a few dried tears.

Next, I would like to thank my supervisor prof. dr. Moens for being interested in my work and providing me useful feedback. I also thank the assessors, for taking the time to read my thesis.

Finally, I would like to thank my family and friends from Lerkeveld. And especially my girlfriend Lieze. What she had to endure this year with me, I would not wish on my worst enemy. Thanks for being there for me, all the time.

Johannes Verherstraeten

Contents

Abstract

State-of-the-art semantic image segmentation methods which are used in autonomous cars, do not make use of temporal relations between consecutive video frames. Each frame is segmented independently from the previous ones. This may result in flickering, when small or narrow objects like poles alternately are and are not detected because of small noise influences. Recurrent neural networks may, unlike feed-forward neural networks, take advantage of the temporal information between consecutive frames to enhance the consistency of semantic segmentation.

More specifically, this thesis first explores some recurrent neural network architectures. By making use of this extra temporal information, they should be able to learn that objects don't continuously appear and disappear in video footage. The effect of different recurrent blocks is tested, together with some hyperparameter variations of these recurrent blocks.

The second aspect is the search for appropriate loss functions. By designing loss functions that punish flickering or favour consistency, we can explicitely encourage these networks to learn temporal consistency relations. Next to the warping loss, this thesis proposes two new loss functions: the change and smoothing loss.

The third aspect is the choice of evaluation metrics. This thesis proposes the temporal Intersection over Union metric (temporal IoU), to assess the temporal consistency of consecutive segmentations.

The final aspect is the creation of a framework that allows to train recurrent neural networks on video files. This framework will be made publicly available when ready.

This thesis achieves an increase of 1.91% on accuracy and an increase of 9.43% on temporal consistency with respect to the baseline network, by adding a convolutional recurrent block and a temporal consistency loss. This is achieved with a loss of execution speed of only 2.9%.

Samenvatting

De methodes voor semantische afbeeldingsegmentatie die momenteel alomtegenwoordig zijn en gebruikt worden in autonome wagens, maken geen gebruik van temporele relaties tussen opeenvolgende videobeelden. Zij segmenteren elk frame onafhankelijk, zonder gebruik te maken van informatie uit vorige frames. Dit kan flikkeringen in de segmentaties tot gevolg hebben, wanneer smalle objecten zoals paaltjes afwisselend wel en niet gedetecteerd worden door invloed van ruis. Het toevoegen van feedbacklussen aan de neurale netwerken die deze segmentaties genereren, geeft hen de mogelijkheid om deze temporele informatie te benutten. Hierdoor kunnen ze de temporele consistentie tussen segmentaties van opeenvolgende videobeelden verbeteren.

Het eerste aspect van deze thesis is het zoeken naar goede architecturen voor neurale netwerken met feedbacklussen. Door gebruik te maken van van de temporele informatie die de feedbacklussen verschaffen, zou zo'n netwerk moeten kunnen leren dat objecten niet continu verschijnen en verdwijnen in videobeelden. Deze thesis test het effect van verschillende soorten feedbacklussen en enkele configuratievariaties.

Het tweede aspect is het zoeken naar geschikte doelfuncties om deze neurale netwerken te trainen. Door doelfuncties te ontwerpen die flikkering bestraffen of consistentie belonen, kunnen we de netwerken expliciet aansporen om die temporele relaties te leren. Naast de warping-doelfunctie stelt deze thesis twee nieuwe doelfuncties voor: de onveranderlijkheidsdoelfunctie en de afvlakdoelfunctie.

Het derde aspect is de keuze van evaluatiemetrieken. Deze thesis stelt een nieuw metriek voor, namelijk de temporele *Intersection over Union* metric (temporele IoU). Deze metric laat toe om de temporele consistentie tussen opeenvolgende videobeelden te beoordelen.

Het laatste aspect is de creatie van een framework dat toelaat om zo'n neurale netwerken met feedbacklussen te trainen met videobestanden. Dit framework zal, wanneer het volledig af is, online en publiekelijk beschikbaar gemaakt worden.

Deze thesis behaalt een verbetering in correctheid van 1.9% en een verbetering in temporele consistentie van 9.43% ten opzichte van het basisnetwerk, door toevoeging van een feedbacklus en een temporele consistentiedoelfunctie. Dit wordt behaald met een verlies in uitvoeringssnelheid van maar 2.9%.

List of Figures and Tables

List of Figures

List of Tables

List of Abbreviations and Symbols

Abbreviations

CNN	Convolutional Neural Network
ReLu	Rectified Linear Unit
CRF	Conditional Random Field
RNN	Recurrent Neural Network
LSTM	Long Short-Term Memory
ConvLSTM	Convolutional Long Short-Term Memory
GRU	Gated Recurrent Unit
ConvGRU	Convolutional Gated Recurrent Unit
IoU	Intersection over Union
fps	Frames per second

Symbols

This list contains the most important symbols that frequently occur in this text and their meaning.

I_t	The input image at time t. Tensor shape: $(3, height, width)$.
O_t	The network output at time t. Tensor shape: $(\#classes, height, width)$.
Y_t	The ground truth segmentation at time t. Tensor shape: $(height, width)$.
$\hat{Y}_t$	The predicted segmentation at time t, derived from the network output. Tensor shape: $(height, width)$.
$\phi_{t \to t-1}$	The dense optical flow between two images I_t and I_{t-1}, in the reference frame of image I_t. Tensor shape: $(2, height, width)$. Can be broken down into $\phi_{h,t \to t-1}$ and $\phi_{w,t-1 \to t}$ for the flow in the vertical and horizontal direction respectively.
C_t	Cell state of a ConvLSTM block at time t. Tensor shape: $(\#channels, h_{state}, w_{state})$
H_t	Hidden state, and at the same time the output of a ConvLSTM or ConvGRU block at time t. Tensor shape: $(\#channels, h_{state}, w_{state})$

Chapter 1

Introduction

1.1 Problem context

The automation of increasingly complex tasks, requires equally increasingly intelligent algorithms. An important part of this intelligence is to be able to see and understand the environment. An example of such an autonome machine is a "cobot" or collaborative robot. As this robot interacts with humans in uncontrolled environments, important measures must be taken to guarantee safety. It continuously scans the environment with cameras, registering the position and movements of nearby objects. Meanwhile, alternative paths are calculated to avoid possible collisions. Similarly, cars get more and more autonomous functions. Currently, cars with Level 2 autonomy are being sold. This means that the computer takes full control of the car, and the driver does not need to hold the steer anymore. It is of paramount importance that such a car can understand its environment and process it in real-time.

The main source of this environmental information are cameras. Since cameras are accurate and cheap nowadays, visual data is abundant. The hard part however, is to design algorithms that can extract meaningful information out of this data. The fact that about half of our human neural tissue is (in)directly related to visual learning [69], suggests that this is not an easy task. The computer vision research domain focuses on how to make computers understand this visual data.

A lot of information can be retrieved from visual data. For example, spatial information and depth can be estimated by using images taken from different viewpoints. In the case of two viewpoints, where the cameras are located next to each other, the technique is called stereo vision. Clearly, inspiration is often found in our human visual system. Another important visual information type is semantic information. Semantic information gives answers to the questions: "What can be seen in the visual data, and where?". In order of increasing detail of localization, we have:

- Image classification: the whole image is associated with a label.
- Object detection: objects in the image are associated with a bounding box and a label.

- Semantic segmentation: each individual pixel of the image is associated with a label.

This thesis focuses on semantic segmentation in the context of autonomous cars. The input will be videos from a camera mounted on the front of a car. In each video frame, the different classes of objects must be detected and localized with pixel-wise accuracy. Since the visual data must be processed in real-time, execution speed is an important limitation in this context. In practice, this means that the segmentation algorithm will be less accurate than the state-of-the-art methods, but will be a lot faster.

The algorithms that will be used in this thesis, belong to the class of convolutional neural networks. Since their large breakthrough in 2012, convolutional neural networks (further abbreviated as CNNs) have made an overwhelming advance and are now the *de facto* standard for extracting semantic information from visual data [17][16][42]. Again, this type of algorithms is based on our basic understanding on how the human brain works. Artificial neurons interconnect to a large network that is able to learn from examples. All state-of-the-art image processing algorithms nowadays use convolutional neural networks.

But while a lot of research has been done on semantic segmentation of images, little work focuses on semantic segmentation of video data. The main reason for this is twofold. First of all, segmenting images is easier. As videos are sequences of temporally related images, they are more complex to process. Secondly, most standard quality metrics for semantic segmentation are image-based. This does not encourage research for better video processing methods. The standard method to segment video data, is to segment each video frame independently, by forwarding them through a feed-forward CNN one by one. The network then sees these frames as unrelated images, and will predict a segmentation without using information from previous frames. However, videos are not sequences of unrelated images. Objects may be classified differently, taking into account their motion or their appearance in the previous frames. Furthermore, there is a lot of redundant information in these image sequences. For example, a pedestrian in the center of a video frame, will not fully disappear in the next frame, assuming the frame rate is high enough. All this information is not leveraged by the standard video segmentation method.

Especially smaller networks, the ones that are used in real-time systems, could benefit from this temporal information. They have the disadvantage that their predictions typically suffer from flickering, when small or narrow objects alternately are and are not detected because of small noise influences. Figure 1.1 visualizes this problem. To improve the temporal consistency of semantic segmentations created by small networks, this thesis explores the use of recurrent layers in the neural network architecture.

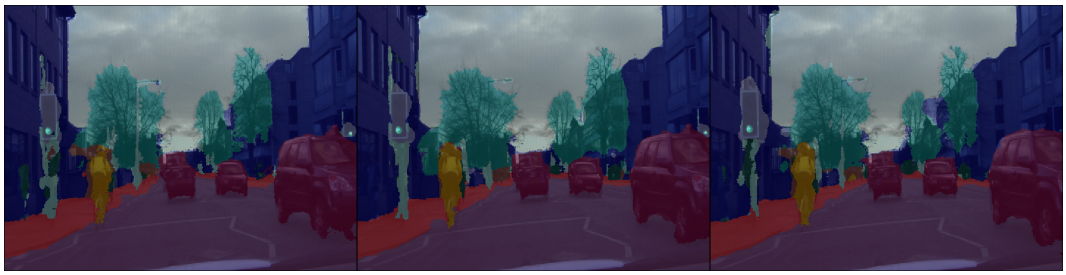

Figure 1.1: Segmentation of a video sequence from the CamVid dataset [6][7] produced by ENet [63]. The lamppost in the center is not detected well in every frame, causing flickering of the segmentation. Also the detection of the building at the right-hand side between the two trees and of the region around the cyclist is flickering.

1.2 Thesis overview

The goal of the thesis is to enhance the temporal consistency in real-time semantic video segmentation, using recurrent neural networks. This research can be divided in four main aspects.

The first aspect is the exploration of convolutional neural network architectures with recurrent connections. These recurrent feedback loops allow information of previous inputs to persist. By making use of this extra temporal information, the network should be able to learn temporal relations in the data, like the fact that objects don't continuously appear and disappear, and therefore learn to avoid the flickering. As a side effect, the network may even learn spatio-temporal features that allow for better classification. For example, an object that is moving is more likely to be a pedestrian than a pole. These features may possibly enhance the classification accuracy of the network. However, in this thesis, the focus is mainly on the enhancement of temporal consistency. Specifically the combination of real-time networks with convolutional RNNs to enhance temporal consistency is, to our knowledge, not yet described in the literature.

The second aspect is the search for appropriate loss functions. A standard semantic segmentation loss function encourages a network to produce segmentations that are as accurate as possible. If the network contains recurrent connections, this may include that the network implicitly learns some notions of temporal consistency. However, by designing loss functions that punish flickering or favour consistency, we can explicitly encourage the network to learn temporal consistency relations.

The third aspect is the choice of evaluation metrics. Most standard segmentation quality metrics are image-based. There is no standard metric yet to assess the temporal consistency of consecutive segmentations. We propose to modify the widely-used Intersection over Union metric, so that it fits this requirement. We also use it to evaluate the consistency of our results, and compare it with the consistency of the baseline cases.

The last aspect is the creation of a framework that allows to implement and test the previous aspects. Most research on deep learning works with text or image data. Both types of data can easily be loaded into memory at once. However, videos are much larger and must be loaded frame by frame. As a consequence, the standard dataloader implementations don't work for video data and we had to implement them ourselves. Another consequence is that the recurrent layers and temporal loss functions and -metrics need to keep a state during the processing of the input frames. These states must be managed well during optimization and evaluation. A large part of the time of this thesis went into the creation and finetuning of this framework.

These aspects lead to the main contributions of this thesis:

- A limited listing of existing recurrent CNN architectures for video processing. Different types of recurrent layers and multiple parameter variations are tested for their influence on execution speed and temporal consistency.
- The proposition of one existing and two new loss functions for temporal consistency: the change, smoothing and warping loss. The warping loss was proposed by Lai et al. [43]. The change and warping loss are both extensively evaluated.
- The proposition of a new evaluation metric for temporal consistency of segmentations: the temporal IoU.
- The implementation of a framework for semantic video segmentation, which is now publicly available at https://github.com/JohannesVerherstraeten/semantic-video-segmentation.

1.3 Thesis structure

This thesis is structured as follows. Chapter 2 provides an overview of the research that already has been done related to this topic. Since the computer vision field is exploding the last few years, an extensive literature study is required to not reinvent the wheel. Chapter 3 becomes more technical and provides a background in recurrent neural networks. As standard neural networks are nowadays widely used in computer vision and other domains, they are assumed to be familiar to the reader. However, recurrent neural networks are not common in computer vision, therefore they are introduced in a separate chapter. Chapter 4 provides a clear definition of the problem, and the proposed solutions to this problem. Chapter 5 then discusses the actual implementation and evaluates the achieved results. Finally, chapter 6 summarizes the results and conclusions, discusses the limitations of our work and proposes further research possibilities. Two additional appendices are also added: appendix A contains the training plots for all networks used in the experiments, and appendix B shortly describes my prior knowledge of this topic and what I did and didn't implement myself.

Chapter 2

Related Work

The computer vision field has been dominated by CNNs since Ciresan et al [17] [16] and Krizhevski et al. [42] showed its power on multiple computer vision benchmarks. This chapter provides an overview of the most important work relevant to this thesis. The first section presents some very important challenges and breakthroughs in deep learning for computer vision. Each of the other sections focuses on a specific subdomain in the computer vision field that provides useful background for this thesis. The subfields that are discussed, in the order of appearance, are: semantic segmentation, recurrent neural networks, action recognition, video object segmentation, semantic video segmentation and temporal consistency.

2.1 General convolutional neural networks

While the theoretical basics for deep learning were already introduced in the second half of the 20th century [22] [44], the breakthrough for computer vision came in 2012 when fast GPU implementations became available. In 2012, Ciresan et al. [17] [16] presented efficient deep convolutional neural networks and greatly improved the records on handwritten digit- and letter recognition (MNIST and NIST) [45] [31], traffic sign recognition [78], generic object recognition (NORB) [46], and many more. A few months later, Krizhevski et al. [42] won the ImageNet competition [19] with their CNN architecture called AlexNet. From then, large breakthroughs followed each other quickly, with mainly one common thread: allowing the increase of the number of layers. Simonyan et al. [75] developed VGG Net, a network with up to 19 layers, and Szegedy et al. [81] developed GoogLeNet/Inception, with up to 22 layers. Both were important milestones in the state-of-the-art perfomance.

Deeper neural networks have a few main challenges. First there is the problem of overfitting. Overfitting, as defined by the Oxford English Dictionary [60], is *"the production of an analysis which corresponds too closely or exactly to a particular set of data, and may therefore fail to fit additional data or predict future observations reliably"*. This problem is not inherent to deep learning only, and usually occurs when the expressivity of the model is much larger than needed to describe the data. In that case, the network doesn't generalize what it sees in the training data, but

learns the training data itself. Some important techniques to reduce overfitting are regularization, cross-validation and dropout [76].

Second there is the problem of vanishing gradients. This problem occurs during the backpropagation step, when the gradient magnitudes in earlier layers become so small that their weights are not updated anymore. As a consequence, the network does not learn anymore. It does not only occur in feed-forward networks with many layers [25], but also in recurrent networks [3] [62] since they can be *unfolded* to deep feed-forward networks. Pascanu et al. [62] show the relation of the problem with the eigenvalues of the recurrent weight matrix. Important techniques to reduce the vanishing gradients problem are normalization layers [37], special activation functions like the (leaky) ReLU [26] and in the case of recurrent neural networks: the LSTM [33] and GRU [14]. Similar to the vanishing gradients problem, recurrent neural networks may suffer from exploding gradients. This is further explained in section 2.3.

Finally, there is the degradation problem. It is shown that deeper networks have exponentially more expressivity than shallow networks [55]. However, increasing depth eventually leads to a decrease of accuracy, independent of overfitting [32] [77]. Some techniques have been developed to overcome this problem. In Highway Networks by Srivastava et al. [77], information can bypass multiple layers, by use of learned gating mechanisms. ResNet, developed by He et al. [32], enhances VGG Net with a similar idea. Here, information can also bypass multiple layers, and is merged with the layer outputs afterwards. These shortcut connections allow networks with up to 150 layers to be trained well.

2.2 Semantic segmentation

The goal of semantic segmentation is to provide a pixel-level labeling of the input. One of the main problems of general neural networks in the context of semantic segmentation, is that the downsampling and pooling layers, providing more abstraction when advancing through the network, also diminish the resolution of the feature maps. To have an output labeling with the same resolution as the input, a couple of different solutions are proposed.

The first solution is to add upsampling layers after the feature maps. Long et al.[51] introduced fully convolutional neural networks for semantic segmentation that could be trained end-to-end. Their decoder uses deconvolutional layers with skip connections to upsample the lower resolution feature maps back to the input size. A lot of enhanced architectures have been designed since. In U-Net, Ronneberger et al. [68] use a larger number of feature channels in the decoder, so that the decoder has more context information available. This makes the number of feature channels in the decoder almost equal to the ones in the encoder, leading to a symmetrical U-shaped architecture. More variants are introduced by Noh et al. [58] (DeconvNet), Badrinarayanan et al. [1] (SegNet) and Lin et al. [50] (RefineNet), building on

improvements in the general deep learning architectures. Peng et al. [64] proposed an encoder-decoder architecture with very large kernel convolutions.

Another solution for the diminished resolution problem is to use dilated convolutions, also known as atrous convolutions. This type of convolution still reduces the resolution, but can leverage a larger receptive field by using sparse filters. Therefore, it can provide more spatial abstraction with the same loss of resolution. Systematic use of these sparse filters support exponential expansion of the receptive field. Although dilated convolutions were originally developed by Holschneider et al. [34] for efficiently computing the undecimated discrete wavelet transform, Chen et al. [10] [12] [11] started using it with success in their DeepLab architectures. Other authors used dilated convolutions in their architectures as well, like Yu et al. [87], Zhao et al. [88] (PSPNet) and Mehta et al. [54] (ESPNet). PSPNet and ESPNet both use an extra technique to aggerage even more large-scale context information. They use spatial pyramid pooling to create multiple scales of a feature map, so that the following layer has access to the information at multiple resolutions. The smaller such a scaled feature map, the more it provides global contextual information to the next layer.

Chen et al. [10] [12] also use a Conditional Random Field (CRF) as a separately tuned post-processing step. It refines a coarse segmentation result using color contrast information from the input image. It can be implemented using classical methods [41] as well as using neural networks [89]. A disadvantage of CRFs is its high computational cost.

Instead of achieving high segmentation accuracy, some work focuses on achieving real-time semantic segmentations. Paszke et al. [63] designed ENet, a deep neural network architecture that adapts ResNet [32] for image segmentation. By heavily reducing the number of parameters, it achieves very fast segmentation speeds, with still reasonable accuracy. In ERFNet, Romera et al. [67] introduce a novel layer that uses residual connections and factorized convolutions. ERFNet is not as fast as ENet, but has a remarkably higher accuracy. Mehta et al. [54] created a neural network module based on efficient spatial pyramids of dilated convolutions, called ESP. The full architecture, called ESPNet, achieves both faster segmentation speeds than ENet and slightly better accuracy on the tested datasets. In this thesis, all architectures will be based on ENet [63], because of its simplicity, compactness and segmentation speed. Its simplicity and compactness reduce training time and resources, while segmentation speed is important in the context of autonomous cars.

2.3 Recurrent neural networks

Recurrent neural network layers have been used extensively in natural language processing [86]. By use of recurrent feedback loops, they allow information of previous inputs to persist. This makes them particularly suitable for sequential data. Standard recurrent neural networks however suffer even more from the vanishing gradients problem than feed-forward ones [3]. Indeed, they can be unrolled to multiple copies

of the same network, passing information to the next copy using the recurrent connection. This way, a RNN can be seen as a very deep neural network.

Similar to the vanishing gradient problem, recurrent neural networks may suffer from the exploding gradients problem [3]. It occurs when the gradients become very large during backpropagation. This may happen when the gradients in the recurrent layer accumulate during the backpropagation through time, when training on a large number of timesteps. It can be remedied by rescaling or clipping the gradients at a certain threshold before updating the network weights.

To further reduce these problems, Hochreiter et al. [33] presented the Long Short-Term Memory (LSTM) block. The LSTM is currently the *de facto* standard in handwriting recognition [29], speech recognition [48], neural machine translation [13] [53], and many more language related fields. An alternative and more compact layer was proposed by Cho et al. [14], called the Gated Recurrent Unit (GRU). The performance of the GRU is similar to that of the LSTM, but it has a smaller number of parameters [15].

Both the LSTM and GRU are designed for natural language processing. When applying them unmodified to images, two main issues arise. First, such classic fully connected recurrent blocks don't take spatial connectivity between pixels into account. Second, the number of weights that such a block would need to process a full image, would be very large. To be able to use these recurrent blocks in an image context, a convolutional LSTM and GRU are presented by respectively Shi et al. [70] and Ballas et al. [2]. These convolutional recurrent blocks allow spatiotemporal features to be learned from video data.

2.4 Action recognition

In action recognition, the goal is to recognize and classify the action that occurs in (a part of) a video sequence. It is related to semantic video segmentation, in the way that it also processes video data. The approach of Simonyan et al. [74] is biologically inspired by the two-stream hypothesis [28]. This hypothesis states that the visual pathway in the human brain consists of two main streams: the ventral stream, related to object recognition, and the dorsal stream, related to spatial localization and motion with respect to the viewer. The ventral stream is implemented as a normal static object recognition neural network, and the dorsal stream is implemented as a CNN operating on the optical flow between the images. The final classificiation is then based on the combination of this information. Ng et al. [56] use a similar approach, but propose to do the combination of the two streams with a LSTM block. Ballas et al. [2] introduce the convolutional GRU to learn spatiotemporal video representations. They applied it successfully to action recognition and video captioning. Li et al. [49] use the convolutional LSTM block and add a motion-based attention mechanism to recognize and locate the action in videos.

2.5 Video object segmentation

Since the annual DAVIS challenges [65] [66] [9] started in 2016, video object segmentation attracts a lot of attention. Video object segmentation differs from semantic video segmentation, since it doesn't have to provide a class label to the objects, but just has to locate them. It's also only required to distinguish between a few foreground objects and the background. The video object segmentation task can have a few different settings. There is a semi-supervised setting, where the user has to provide a ground-truth segmentation mask or a scribble mask for the object in the first video frame, and the method has to generate the segmentation masks of the same object in the subsequent frames. There is also an unsupervised setting, where the method itself has to determine the most prominent objects and track them throughout the video. This section only mentions methods that use temporal information.

Jain et al. [38] propose the SegNet architecture that can operate in both settings. It is a two-stream architecture, which means that it has an appearance stream and a motion stream. The appearance stream in such an architecture is often based on an standard object detection network, in this case ResNet-101 [32]. It accepts the video frames one by one. The motion stream, in this case exactly the same network architecture as the appearance stream, accepts optical flow images calculated on pairs of video frames. A fusion module processes the information provided by both streams, delivering the final predictions. Tokmakov et al. [82] also propose a two-stream method, but they add a convolutional GRU to the fusion module to learn a visual memory representation of the scene. It refines the initial estimates of the two streams, by using memory information of the previous frames.

Gadde et al. [23] approach the problem in a different way. To adapt standard CNN models for video data, they propose to use an optical flow warping method (NetWarp) that maps representations of the previous frame to the current frame. Yet a very different approach is presented by Voigtlaender et al. [85]. Their network updates its weights online, to be able to adapt to changes in appearance and better fit the situation.

2.6 Semantic video segmentation

In semantic video segmentation, the goal is to semantically segment each frame using both spatial and temporal features. Fayyaz et al. [21] propose to use a module based on a classic LSTM to learn the temporal characteristics of videos. They first apply a CNN to each frame, to extract spatial features. Then, the results are propagated through the temporal LSTM-based module. Finally these spatio-temporal features are deconvolved to achieve the resulting pixel-wise segmentations. Their result achieves state-of-the-art semantic segmentations. However, the disadvantage of their method is that it uses multiple classic LSTMs, each one working on a part of the spatial domain, which negatively effect the execution speed. The approach of Byeon

et al. [8] is similar. To tackle this issue, Siam et al. [72] and Valipour et al. [84] use convolutional GRUs in their semantic video segmentation architectures, as they were already introduced by Ballas et al. [2] in action recognition and video captioning. Including a single recurrent layer allows the network to remember what it saw in previous frames. However, it does not allow the network to learn motion. To bypass this limitation, Nilsson et al. [57] propose to warp the hidden state of a convolutional GRU along the optical flow between the consecutive frames. This way, its memory of the previous frames moves along with the objects in the scene. In 2018, Siam et al. [71] summarized most of these works in a comparative study of real-time semantic segmentation methods for autonomous driving.

2.7 Temporal consistency

This section lists work that pursues temporal consistency in video data, whether or not in combination with semantic segmentation. In this context, a method is called *blind* if it is agnostic to the specific image processing algorithm applied to the original video. Bonneel et al. [5] propose a method for blind video temporal consistency, based on the minimization of the optical flow warping error between consecutive frames. Lai et al. [43] present an approach based on a ConvLSTM to make per-frame processed videos temporally consistent. They use both long-term and short-term consistency losses alongside a perceptual loss. The short-term temporal loss is defined as the optical flow warping error between the output frames. The long-term temporal loss is defined as the warping error between the first frame and the following 10 output frames. The perceptual loss is based on the similarity of the original output and the temporally consistent output. Their method doesn't need optical flow at test time and achieves real-time speed, even for high-resolution video data. The main recurrent network architecture and the warping loss function in this thesis are respectively based on and taken from their work. We propose however two other temporal loss functions, and a temporal consistency metric specific for video segmentation.

2.8 Conclusion

This thesis focuses on combining the semantic video segmentation methods with recurrent neural networks and temporal consistency methods. Especially small networks that are able to operate real-time are investigated, since they suffer more from flickering and temporal inconsistencies. Inspiration to tackle this problem is also found in the domains of action recognition and video object segmentation, as they also deal with temporal information in the context of image sequences. More specifically, the ENet architecture from Paszke et al. [63] is used extensively in this thesis, as it is a simple and real-time network. All network architectures that will be tested are variants of this ENet. Also the ConvLSTM [70] and ConvGRU [2], the convolutional variants of the LSTM [33] and GRU [14] are used as the recurrent building blocks. The architecture variations that will be described are based on the

ones proposed by Nilsson et al. [57] and Lai et al. [43]. Also, the warping loss, one of the losses that is extensively used in the experiments, is proposed by Lai et al. [43]. This thesis could be seen as an extension of their work, applied in a real-time context. Moreover, it tests multiple architecture variations and proposes two new temporal loss functions and a new temporal evaluation metric. Finally, the idea of a two-stream architecture is not used in this thesis, but could be a promising path for further research.

Chapter 3

Background - Recurrent Neural Networks

A recurrent neural network (RNN) has, in contrast to feed-forward neural networks, at least one feedback loop. The feedback loop feeds output at timestep $t-1$ as additional input at timestep t. This feedback loop allows information of previous inputs to persist, so it is also called the RNN state or memory. One can see a RNN as a sequence of copies of the same feed-forward network, with a copy for each timestep. This is the *unrolled* view of the RNN, and is shown in figure 3.1.

Recurrent neural networks are not often used in the image processing context. Therefore, the following sections provide a background in the most commonly used recurrent building blocks: Long Short-term Memory (LSTM) [33] and Gated Recurrent Unit (GRU) [14]. This part is based on the excellent blogpost by Christopher Olah (Colah) [59]. The last section shortly describes how backpropagation is applied RNN.

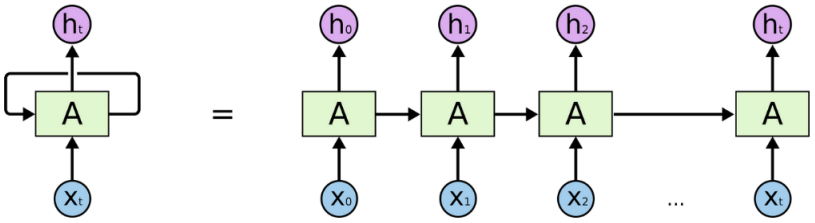

Figure 3.1: A simple recurrent network and its unrolled view. Image copied from Colah's blog [59].

3.1 Long Short-Term Memory

Standard recurrent neural network blocks struggle to learn long-term information, because of the vanishing and exploding gradients problem [3] [62]. Long Short-Term Memory (LSTM) is specialized to reduce these problems. The original version of the

LSTM, proposed by Hochreiter et al. [33] uses fully connected network layers. It has proven to be powerful for sequence modeling with long-term dependencies and natural language processing [48] [13] [53]. This fully connected LSTM is explained in section 3.1.1. For image processing, a much more convenient LSTM variant is proposed by Shi et al. [70]: the convolutional LSTM (ConvLSTM). This variant is explained in section 3.1.2.

3.1.1 Fully connected LSTM

Figure 3.2 shows the configuration of an LSTM. It consists of 2 states, called the cell state c_t and the hidden state h_t. Confusingly, the hidden state is also the output. The cell state contains the long-term information. The LSTM uses gating mechanisms to carefully control the information flow in and out the cell state. More specifically, the LSTM has three different gates. The first gate determines which information from the cell state may be removed at time t, and is called the forget gate f_t.

$$f_t = \sigma(W_f[h_{t-1}, x_t] + b_f) \tag{3.1}$$

Here is σ the sigmoid function, W_f the learned parameters of the forget gate, h_{t-1} the previous hidden state, x_t the current input and b_f a bias term. The second gate determines which information of the candidate hidden state g_t will be added to the cell state at time t, and is called the input gate i_t.

$$g_t = \tanh(W_g[h_{t-1}, x_t] + b_g) \tag{3.2}$$

$$i_t = \sigma(W_i[h_{t-1}, x_t] + b_i) \tag{3.3}$$

The formula for updating the old cell state c_{t-1} to the new cell state c_t is then:

$$c_t = f_t \odot c_{t-1} + i_t \odot g_t \tag{3.4}$$

Finally, the output gate o_t determines what the next hidden state h_t and thus output of the LSTM will be.

$$o_t = \sigma(W_o[h_{t-1}, x_t] + b_o) \tag{3.5}$$

$$h_t = o_t \odot tanh(c_t) \tag{3.6}$$

Some authors use slightly different variants of this LSTM configuration [59]. Gers and Schmidhuber [24] introduced a popular variant with peephole connections. Here, the gates are not only based on h_{t-1} and x_t, but also on c_{t-1} or c_t. The formulas for the gates then become:

$$f_t = \sigma(W_f[c_{t-1}, h_{t-1}, x_t] + b_f) \tag{3.7}$$

$$i_t = \sigma(W_i[c_{t-1}, h_{t-1}, x_t] + b_i) \tag{3.8}$$

$$o_t = \sigma(W_o[c_t, h_{t-1}, x_t] + b_o) \tag{3.9}$$

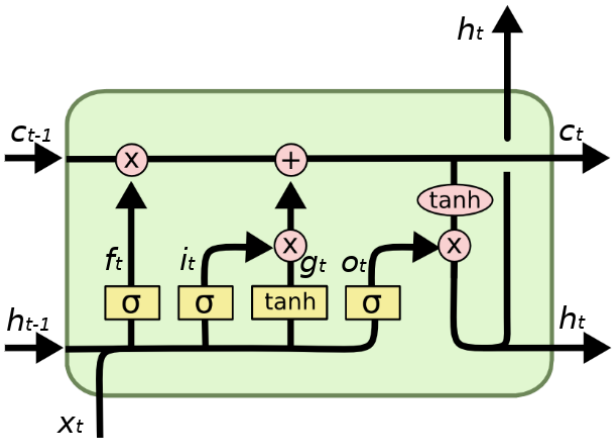

Figure 3.2: The standard LSTM configuration. The top horizontal line is the cell state, the bottom horizontal line is the hidden state. The yellow layers are learned neural network layers with a specified non-linearity, the pink circles are pointwise operations, merging arrows are concatenations, and diverging arrows are copies. Image copied from Colah's blog [59].

3.1.2 Convolutional LSTM

The fully connected LSTM has two issues when applied to image data. First, it doesn't take spatial connectivity between pixels into account. This spatial information however is crucial in images. Second, the number of weights that a fully connected layer would need to process a full image, would be very large. Using convolutional layers for the gates instead of fully connected ones, solves both problems. This convolutional LSTM (ConvLSTM) is proposed by Shi et al. [70]. The difference is that the inputs X_t, outputs H_t, cell states C_t and gates I_t, F_t and O_t are now 3D tensors with dimensions (channels, height, width), and that the matrix multiplication with the network weights is replaced by the convolution operator $*$. An extra hyperparameter is also introduced: the size of the convolution kernel. The formulas for the convolutional LSTM become:

$$F_t = \sigma(W_f * [H_{t-1}, X_t] + b_f) \tag{3.10}$$

$$G_t = \tanh(W_g * [H_{t-1}, X_t] + b_g) \tag{3.11}$$

$$I_t = \sigma(W_i * [H_{t-1}, X_t] + b_i) \tag{3.12}$$

$$C_t = F_t \odot C_{t-1} + I_t \odot G_t \tag{3.13}$$

$$O_t = \sigma(W_o * [H_{t-1}, X_t] + b_o) \tag{3.14}$$

$$H_t = O_t \odot tanh(C_t) \tag{3.15}$$

3.2 Gated Recurrent Unit

The Gated Recurrent Unit (GRU) is a more compact variant of the LSTM. It reduces the number of parameters, and thereby also the execution time, while not compromising on performance [30]. The fully connected GRU is introduced by Cho et al. [14] as an alternative for the LSTM. Figure 3.3 shows the configuration of a GRU. First of all, the cell state and hidden state are merged into one hidden state. Therefore, an output gate is not needed anymore. A second modification with respect to the LSTM, is the addition of a reset gate r_t, which controls what information from the hidden state h_{t-1} will be added to the new candidate hidden state g_t.

$$r_t = \sigma(W_r[h_{t-1}, x_t]) \tag{3.16}$$

$$g_t = \tanh(W_g[r_t \odot h_{t-1}, x_t]) \tag{3.17}$$

The final difference, is that the GRU couples the forget and input gate. When some information is forgotten, new information must be added in its place. The combination of these gates is called the update gate z_t.

$$z_t = \sigma(W_z[h_{t-1}, x_t]) \tag{3.18}$$

The final equation for the output is:

$$h_t = (1 - z_t) \odot h_{t-1} + z_t \odot g_t \tag{3.19}$$

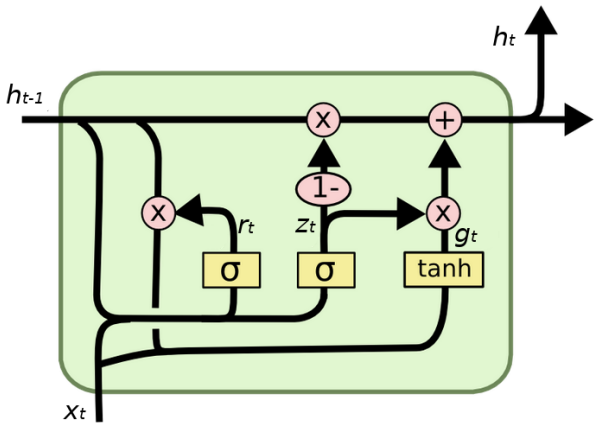

Figure 3.3: The GRU configuration. The top horizontal line is the cell state. The yellow layers are learned neural network layers with a specified non-linearity, the pink circles are pointwise operations, merging arrows are concatenations, and diverging arrows are copies. Image copied from Colah's blog [59].

Similar to the convolutional LSTM, a convolutional variant of the GRU, named ConvGRU, is developed too [2]. For completeness, the equations of the convolutional

GRU are:

$$R_t = \sigma(W_r * [H_{t-1}, X_t]) \tag{3.20}$$

$$G_t = \tanh(W_g * [R_t \odot H_{t-1}, X_t]) \tag{3.21}$$

$$Z_t = \sigma(W_z * [H_{t-1}, X_t]) \tag{3.22}$$

$$H_t = (1 - Z_t) \odot H_{t-1} + Z_t \odot G_t \tag{3.23}$$

3.3 Backpropagation through time

In theory, since information in recurrent neural networks can persist during a whole input sequence, the prediction on the last timestep may depend on the input of the first timestep. This can be seen well in the unrolled network of figure 3.1. Therefore, during training, at every labeled timestep, backpropagation should be performed all the way back to the first timestep. During each pass through the recurrent part, one for each passed timestep, the gradients are updated. This concept is called backpropagation through time (BPTT).

Propagating the loss all the way back to the first timestep is a time-consuming operation, and in case of video data, a memory-consuming operation too. To solve this problem, one could split large sequences in smaller ones and perform backpropagation on each small sequence separately. However, the network would then be blind to temporal dependencies that are longer than the small sequence length [80].

Instead, a truncated version of backpropagation through time is usually performed over the whole sequence [80]. Here, the backpropagation update is only performed for a fixed number of timesteps k_2 and then truncated. Algorithm 1 describes this for a backpropagation that is performed every k_1 timesteps. The advantage of this method is that the hidden states process the whole sequence, while still allowing efficient optimization.

for t **from** 1 **to** t **do**
  Run the RNN for one step;
  if t **divides** k_1 **then**
    Run BPTT from t down to $t - k_2$;
  end
end

Algorithm 1: Truncated BPTT. Algorithm copied from [80].

Note however that the training time might increase heavily when the number of recurrent connections or the number of backpropgatation steps through time increase. For example, imagine a network with one recurrent block such as in image 3.1 being trained with 1 backpropagation step through time. A loss value calculated on h_t will partly backpropagate to input x_t and will partly pass through the recurrent connection and backpropagate to input x_{t-1}. However, when the network has two recurrent blocks placed in series, three such paths exist. Indeed, one path does

not pass though a recurrent connection and arrives at x_t, the two other paths take only the first or the second recurrent connection and arrive at x_{t-1}. The fourth path, taking both recurrent connections would not be taken since it would finish in x_{t-2}, which is two steps back in time. So the first layers of the network are passed three times during loss backpropagation. This number increases heavily when more recurrent blocks or more backpropagation steps through time are used. As a consequence, the training time increases too. Therefore, this thesis limits the number of recurrent blocks and the number of backpropagation steps through time to 1.

3.4 Conclusion

Using feedback loops, recurrent neural networks can allow information of previous inputs to persist. The LSTM and the GRU are two recurrent blocks that use gating mechanisms to control the information flow in and out of the cell state. This gating makes sure that long-term information is not immediately pushed away by new inputs when it still might be relevant. In the following chapters, the convolutional versions of the LSTM and GRU are used. These versions are specifically designed to take spatial connectivity into account and to be more efficient when working with image data. All recurrent networks that are used in the evaluation chapter are trained with 1 recurrent block and 1 backpropagation step through time.

Chapter 4

Methodology

This chapter is the main theoretical body of the thesis. First, it provides a clear definition of the problem and its subproblems. It also introduces the variable naming conventions used throughout the rest of the thesis. Thereafter, section 4.2 discusses the different network architectures that are investigated in this thesis. Section 4.3 describes four loss functions. The first one, the cross-entropy loss, is a standard semantic segmentation loss function. The three other ones are temporal consistency loss functions, that may encourage the networks to learn temporal consistency. Finally, section 4.4 proposes the metrics that will be used to evaluate the quality of the results.

4.1 Problem definition

The goal of this thesis is to enhance the temporal consistency between semantic segmentations of consecutive video frames, using recurrent neural networks, in the context of autonomous cars. First of all, we search for network architectures that allow to use the temporal information in video data. For this, we will use recurrent convolutional neural networks. The hypothesis is that having access to this temporal information will improve the temporal consistency. In addition, we test multiple loss functions to encourage the network to deliver temporally consistent results.

The input of the network will be a video dataset with street scenes. A video consists of frames I_t with dimensions $(3, H, W)$, where H and W stand for height and width, and 3 for the RGB channels. Each video must have at least one frame with a ground truth semantic segmentation map (also called label) Y_t. The labels Y_t must be matrices with dimensions (H, W), with for each pixel an integer value i encoding a semantic class. Each value i must be in range $[0, C)$, where C is the number of semantic classes in the dataset. The network must be applicable to multiple datasets, which each may have a different number of semantic classes C and different encodings. However, only one dataset will be used at a time, and the encoding within one dataset must be consistent. That is: a value i representing the class *tree* in one label, must represent *tree* in all other labels too. The dataset will be divided in a train, validation and test set. The network will be trained on

the train dataset. To prevent the network from overtraining, it will be regularly validated on the validation dataset. The network state with the highest accuracy on the validation dataset will be kept. This network is finally evaluated on the test set.

The output tensor O_t of the system will be a tensor with shape (C, H, W) for each input video frame I_t. The values in the tensor encode the likelihood that the pixel on position (h, w) belongs to class i. An output label $\hat{Y}_t$, where exactly one class is assigned to each pixel, can be found by taking the class indices with the highest likelihood for their pixel:

$$\hat{Y}_t[h, w] = \underset{i}{\operatorname{argmax}}\ O_t[i, h, w] \tag{4.1}$$

The output will have the same class encoding as the training dataset. If a ground truth label Y_t is available for the input frame, the output label $\hat{Y}_t$ should be as similar as possible to it. Besides, the output labels $\hat{Y}_t$ of consecutive input frames should be temporally consistent. This means that if a certain object is detected in one frame and it is still visible in the next one, it should be detected there as well and vice versa. Figure 4.1 visualizes the relations between the input, output and predicted label.

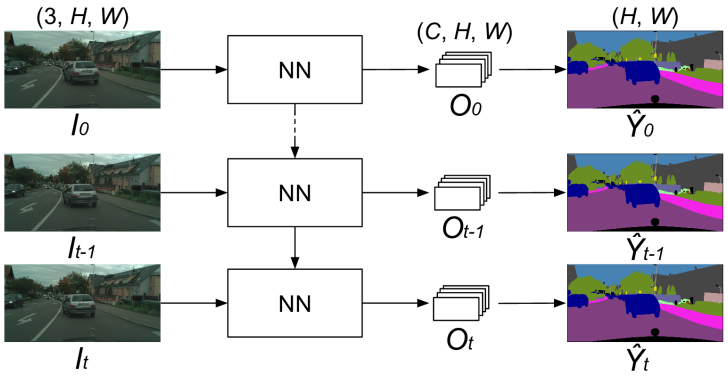

Figure 4.1: Overview of the system. $I_{0..t}$ are the video input frames. NN is a neural network with at least one recurrent connection. $O_{0..t}$ are the network output tensors. $\hat{Y}_{0..t}$ are the predicted labels, which can be obtained from the outputs with equation (4.1).

The system will be a fully convolutional neural network. To be able to use temporal information, at least one recurrent connection will be present in the architecture. The second objective, after enhancing temporal consistency, is to have an algorithm that can process a video in real-time. This is an important requirement to be useful in autonomous cars. To meet this requirement, the network will have to be small in comparison with state-of-the-art semantic segmentation networks. Other aspects of the network may have to be optimized too.

The training of the system requires a loss function L to be defined. During training, the system is modified in such a way that it minimizes the loss on the

training dataset. A standard loss function for semantic segmentation is the cross-entropy loss. The more the output O_t of the network deviates from the corresponding ground truth label Y_t, the larger this loss becomes. We will test multiple additional loss functions, to train the network so that it delivers temporally consistent results.

4.2 Network architectures

There are a lot of successful neural network architectures around. This thesis uses ENet [63] as the base architecture. It is a lightweight and fast feed-forward network, and is used in mobile and real-time applications. Since there is always a tradeoff between speed and accuracy, its accuracy is not state-of-the-art. Such fast but less accurate networks generally have less temporally consistent results. The architecture of a feed-forward network also doesn't allow it to learn temporally consistency, since no information is shared between different timesteps. Yet there is a lot of redundant information in consecutive video frames. Street scenes don't change drastically in 0.1 seconds. However, the current standard is to start segmenting each single frame from scratch. The fact that they don't leverage this continuity-information, can be seen as a waste.

This section proposes some ENet-based architectures that may learn this temporal information. By use of recurrent connections, they can pass through information to following timesteps. It can be seen as a form of memory: the networks remember what they saw in previous frames, and can use this information to better segment the current one.

4.2.1 ENet

As the following architectures will be based on ENet by Pazske et al. [63], this network is described first. ENet is a small network, meant for real-time semantic segmentation. It is an encoder-decoder network, consisting of 29 modules. The first module is the ENet initial block, shown in figure 4.2b. The intermediate modules are ENet "bottlenecks", shown in figure 4.2b. They use a residual connection, based on ResNet [32]: they have a main branch, propagating the data with only few modifications, and an extension branch with convolutional filters. The last module is a full convolution (also known as deconvolution) layer. Figure 4.3 visualizes the configuration of these modules.

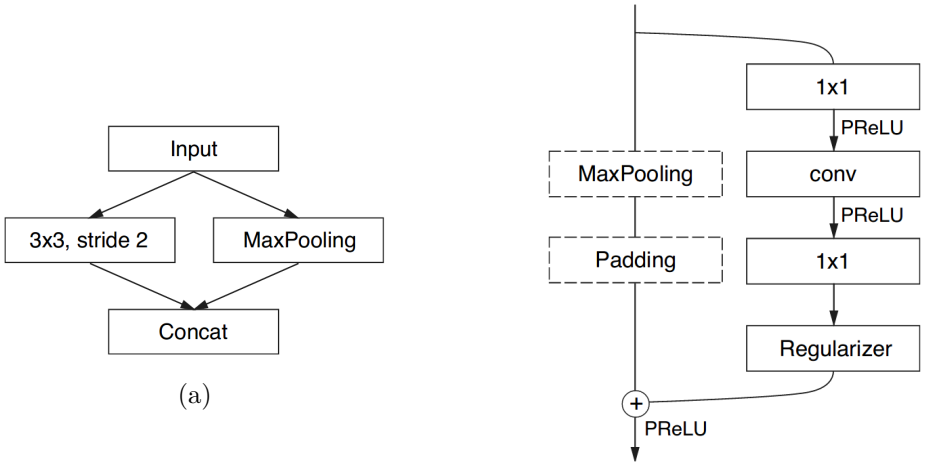

Figure 4.2: (a) The ENet initial block. (b) The ENet bottleneck module. The *conv* block is either a regular, dilated, or full convolution (also known as deconvolution). The *regularizer* block performs Spatial Dropout [83]. Images copied from original ENet paper [63].

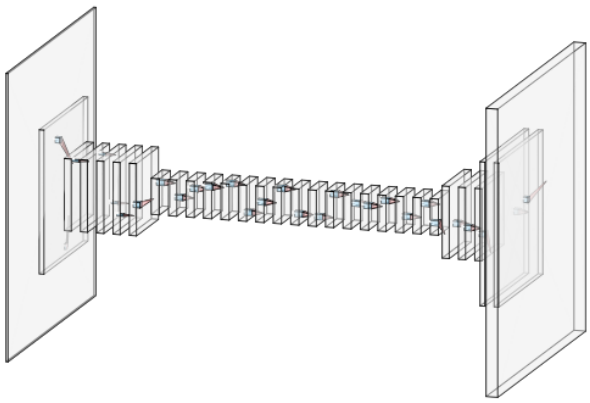

Figure 4.3: Visualization of the ENet architecture. Layers visualized here are actually blocks as shown in figure 4.2, not standard convolutional layers. Image created using this tool [47].

4.2.2 ENet + appended ConvGRU

Nilsson et al. [57], propose to append a convolutional Gated Recurrent Unit (ConvGRU) to a standard semantic segmentation network. The ConvGRU feeds its output segmentation of the previous frame as input to itself. But first, this segmentation of the previous frame is warped by a Spatial Transformer block, to better fit the current timestep. So the ConvGRU receives both the current segmentation and the warped previous segmentation, and learns to combine them optimally. A schematic

of this configuration is shown in figure 4.5.

The Spatial Transformer block makes use of externally calculated dense optical flow. Dense optical flow $\phi_{t\to t-1}$ between two images I_t and I_{t-1} gives for each pixel in image I_t the translation to the the pixel representing the same point of the scene in image I_{t-1}. Figure 4.4 visualizes the optical flow. It can be represented by the following formula:

$$I_t[h, w] = I_{t-1}[h + \phi_{h,t\to t-1}[h, w], y + \phi_{w,t\to t-1}[h, w]] \tag{4.2}$$

where $\phi_{h,t\to t-1}$ denotes the translation in the vertical direction and $\phi_{w,t-1\to t}$ the translation in the horizontal direction. The calculation of dense optical flow can be done in multiple ways [52] [36], but there will always be incorrectness because of occlusion or deformation happening between the frames, or because of the aperture problem [4]. This aside, equation (4.2) allows to define optical flow warping:

$$I_t \sim I_{t-1,warped} = \text{warp}_{t-1\to t}(I_{t-1}) \tag{4.3}$$

Here, image I_{t-1} is warped foward in time to be similar to I_t. Note that this forward warping function uses the backward flow $\phi_{t\to t-1}$, as can be seen in equation (4.2). The Spatial Transformer uses optical flow warping to warp the hidden state of the previous timestep to the current timestep.:

$$H_{t-1,warped} = \text{warp}_{t-1\to t}(H_{t-1}) \tag{4.4}$$

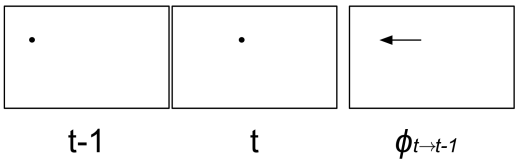

Figure 4.4: Visualization of the optical flow between two consecutive frames. The backward optical flow is used for forward warping.

The proposed ConvGRU is also adapted with respect to its standard implementation: the reset gate R_t, which controls what information from the previous hidden state H_{t-1} will be added to the new candidate hidden state G_t, is not learned anymore. Instead, it uses a confidence measure of the optical flow. Only if the optical flow in a certain region of the image is probably accurate, the previous hidden state information of that region is allowed to pass through. Such a confidence map is achieved by subtracting the current input image I_t with the optical-flow-warped previous input image $I_{t-1,warped}$. The equation for the reset gate becomes:

$$R_t = 1 - tanh(|W_r * (I_t - I_{t-1,warped} + b_r|) \tag{4.5}$$

The proposed ConvGRU has some other modifications with respect to the standard ConvGRU, but they are minor and are not in the scope of this text. The full model is differentiable and therefore trainable end-to-end.

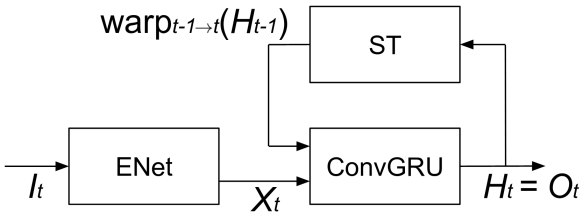

Figure 4.5: Schematic of ENet + appended ConvGRU. I_t is the input frame at timestep t, and H_t is the corresponding output segmentation.

4.2.3 ENetGRU / ENetLSTM

The ENet + appended ConvGRU architecture depends on the availability of dense, qualitative optical flow information. The optical flow almost directly influences the output of the network, since it is injected just before the final ConvGRU block. Therefore, the quality of the optical flow is important for a qualitative netwerk output. Also, the calculation of dense optical flow is time-consuming. This makes it less interesting in a real-time system. Therefore, it is not evaluated in the evaluation chapter.

To overcome the need of optical flow, one could move the ConvGRU block to the center of the ENet, between the encoder and the decoder. The resolution in the center layers of ENet is 8x smaller than the resolution of the output. Because the spatial compression is much larger in the center layers, small movements of objects in the scene have less influence on the local representation. Therefore, if the center layers use information of previous frames, it is less important to spatially adjust this information to better fit the current frame. Valipour et al. [84] also use this idea in their network for video object segmentation. Lai et al [43] use it too in a network for real-time blind video temporal consistency. A schematic of this configuration is shown in figure 4.6a. In the following chapters, this network will be referred to as EnetGRU.

Of course, the same architecture could be used with a ConvLSTM: this variant will be referred to as ENetLSTM. Other variants of this architecture include a residual connection to bypass the ConvGRU. This residual connection can be added or appended to the ConvGRU output, as in figure 4.6b. The experiments will compare these variants.

4.3 Loss functions

During training, the network is modified to minimize a certain loss function L. This loss function determines what the network should learn. The standard loss function for semantic image segmentation is the cross entropy loss L_{CE}. This loss encourages the network to produce labels $\hat{Y}_t$ that are as close as possible to the ground truth labels Y_t. This loss is explained in section 4.3.1. To learn the network to generate more consistent segmentations of consecutive video frames, a temporal consistency

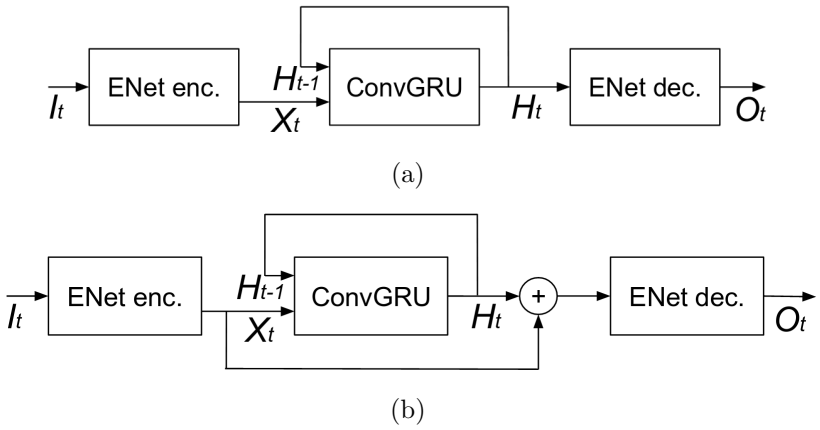

Figure 4.6: (a) Schematic of ENetGRU. (b) Schematic of ENetGRU, with addition of a residual connection. This connection can be added or appended to the ConvGRU result. In both images, the *ENet enc* and *ENet dec* are respectively the encoder and decoder part of the ENet architecture. The ConvGRU can be replaced with a ConvLSTM to form the ENetLSTM architecture.

loss function L_{TC} could be added to this standard loss function. Sections 4.3.2 to 4.3.4 each propose a different loss function that may improve the temporal consistency. The change loss and smoothing loss are new loss functions proposed in this thesis, the warping loss already existed and was proposed by Lai et al. [43]. The final loss functions that are used in the experiments are a weighted sum of the cross entropy loss and one of the temporal consistency losses:

$$L = \alpha L_{CE} + \beta L_{TC} \tag{4.6}$$

4.3.1 Cross entropy loss

The cross entropy loss, also known as logistic loss, accepts two probability distributions $p(x)$ and $q(x)$. It is defined as:

$$L_{CE} = -\sum_{\forall x} p(x) \log(q(x)) \tag{4.7}$$

In the context of semantic image segmentation with neural networks, it is used to grade a prediction O_t of the network, given the ground truth Y_t. Note that the predicted output O_t of the network has shape (C, H, W), with C the number of semantic classes. If an image I_t is provided, the network gives for each pixel (x, y) a vector of C scores: $\hat{p}$:

$$\hat{p} = [\hat{p}_0, \hat{p}_1, ..., \hat{p}_{C-1}] = O_t[:, x, y] \tag{4.8}$$

Each score $\hat{p}_i$ indicates how likely the pixel belongs to class i. Assume that the scores $\hat{p}_i$ for each pixel summate to 1. In that case, the vector $\hat{p}$ can be seen as a

probability distribution over $[0, C)$. The ground truth label Y_t corresponding to the image I_t, provides for each pixel the correct class $i_{correct}$. This can also be seen as a probability distribution p over $[0, C)$, where all probabilities p_i are 0, except for the probability of the correct class $p_{i_{correct}} = 1$. This is called a one-hot encoding of the class. If we put both probability distributions in equation (4.7), we get the cross entropy loss for the prediction of a single pixel:

$$L_{CE} = -\sum_{\forall i} p_i \log(\hat{p}_i) = -\log(\hat{p}_{i_{correct}}) \tag{4.9}$$

However, the scores for a pixel don't generally summate to 1. To make sure they do, a Softmax function can be applied to the scores first. The Softmax function is defined as:

$$f(p)_i = \frac{e^{p_i}}{\sum_j^C e^{p_j}} \tag{4.10}$$

It normalizes a vector of real numbers into a probability distribution. That is, every element will be rescaled to a value in the range $[0, 1]$, and all values will summate to 1. Applying the Softmax function before the cross entropy loss gives:

$$L_{CE} = -\log\left(\frac{e^{\hat{p}_{i_{correct}}}}{\sum_i^C e^{\hat{p}_i}}\right) \tag{4.11}$$

This last formulation is also refered to as the *categorical* cross entropy loss [27]. The cross entropy loss for a full image is obtained by averaging the losses of all its pixels. Together with a substitution of equation (4.8) and the fact that for each pixel (x, y) applies that $Y_t[x, y] = i_{correct}$, the categorical cross entropy loss for a full image is:

$$\mathrm{L_{CE}}\left(Y_t, O_t\right) = -\frac{1}{HW}\sum_{x,y}\log\left(\frac{e^{O_t[Y_t[x,y],x,y]}}{\sum_i^C e^{O_t[i,x,y]}}\right) \tag{4.12}$$

4.3.2 Change loss

The change loss is a temporal consistency loss function that penalizes the differences between consecutive predictions O_{t-1} and O_t. It can be seen as the ultimate temporal consistency loss. A network trained with this loss only, will generate the same prediction for each frame of a video. The change loss generates a label $\hat{Y}_{t-1}$ from the previous network prediction O_{t-1}, using equation (4.1):

$$\hat{Y}_{t-1} = \underset{i}{\operatorname{argmax}}\ O_{t-1}[i] \tag{4.13}$$

and encourages the current prediction to be equal using the (categorical) cross entropy loss:

$$\mathrm{L_{change}}(O_{t-1}, O_t) = \mathrm{L_{CE}}\left(\hat{Y}_{t-1}, O_t\right) \tag{4.14}$$

4.3.3 Smoothing loss

The smoothing loss is also a temporal consistency loss, but is less strict than the change loss. It is more tolerant to small motions in the predictions. For example: if a pedestrian walks in the scene, its segmentation should move along in the image. Consecutive segmentations of the pedestrian will not be the same, but should lie close to each other. To allow such small motion to happen, the smoothing loss spatially smooths the prediction at the previous timestep, before comparing it to the current prediction.

The smoothing loss is calculated as follows. First, apply the Softmax function to the outputs O_{t-1} and O_t for normalization. Define $\hat{P}_{t,i}$ as the probability map of class i in the output O_t:

$$\hat{P}_{t,i} = O_t[i] \tag{4.15}$$

The probability map of the previous timestep smoothed is:

$$\hat{P}_{t-1,i,smoothed} = \hat{P}_{t-1,i} * G_{\sigma_x,\sigma_y} \tag{4.16}$$

where G_{σ_x,σ_y} is a Gaussian kernel. The output $O_{t-1,smoothed}$ is reconstructed as follows:

$$O_{t-1,smoothed}[i] = \hat{P}_{t-1,i,smoothed} \tag{4.17}$$

Afterwards, a Softmax function is again applied to the resulting $O_{t-1,smoothed}$ for normalization. Finally, the smoothing loss is then defined as the inverse of the mean of the pointwise product of $O_{t-1,smoothed}$ and O_t:

$$\mathrm{L}_{\text{smooth}}(O_{t-1}, O_t) = \frac{1}{\frac{1}{CHW}\sum_{i,h,w} O_{t-1,smoothed}[i,h,w]O_t[i,h,w]} \tag{4.18}$$

4.3.4 Warping loss

The smoothing loss teaches a network that motion may occur in all directions equally likely. If an object moves to the left, but the network falsely predicts that it moves the same amount of distance to the right, the smoothing loss will be similar to that of a correct prediction. The warping loss tries to avoid this. Lai et al. [43] propose such a warping loss function for temporal consistency. Their loss function warps the previous output O_{t-1} along the optical flow $\phi_{t,t-1}$, and computes the difference with the current output O_t. First, the previous input image I_{t-1} and the previous output O_{t-1} are warped along the optical flow using equation (4.3):

$$I_{t-1,warped} = \text{warp}_{t-1,t}(I_{t-1}) \tag{4.19}$$

$$O_{t-1,warped} = \text{warp}_{t-1,t}(O_{t-1}) \tag{4.20}$$

Then, the warping correctness mask $M_{t-1,t}$, also the called visibility mask, is calculated. It makes sure that incorrect optical flow doesn't influence the loss.

$$M_{t-1,t} = \exp\left(-\alpha \|I_t - I_{t-1,warped}\|_2^2\right) \tag{4.21}$$

Here, α is a scaling parameter that the authors [43] empirically set to 50 for output values in range [0, 1]. Finally, the warping loss between two consecutive outputs is defined as:

$$\mathrm{L}_{\mathrm{warp}}(O_{t-1}, O_t) = \sum_{i,h,w} \left(M_{t-1,t} \| O_t - O_{t-1,warped} \|_1\right)[i,h,w] \tag{4.22}$$

4.4 Evaluation metrics

The segmentations produced by our methods need to be evaluated in an objective way. This allows them to be compared to other methods. Semantic image segmentation results are often evaluated using the Intersection over Union metric (IoU). This metric compares a predicted label to the ground truth label, and is explained in the first section. However, this metric does not measure temporal consistency. We will measure the temporal consistency between two consecutive segmentations using a new metric: the temporal Intersection over Union. This is explained in the second section. Finally, since execution in real-time is important in autonomous cars, the execution speed of the methods is measured.

4.4.1 Intersection over Union

The Intersection over Union (IoU) metric, also known as the Jaccard index, measures the accuracy of a predicted label by comparing it to the ground truth. It is independent from the algorithm that produced the label, and is therefore widely used in object detection and segmentation challenges like the DAVIS challenge [66] [9] and PASCAL VOC [20].

The IoU is defined as the area of the intersection, divided by the area of union of the predicted and the ground truth detection:

$$\mathrm{IoU}(P, G) = \frac{|P \cap G|}{|P \cup G|} \tag{4.23}$$

Its value lies between 0 (no overlap) and 1 (perfect overlap). In multi-class semantic segmentation, an IoU value is calculated for each single class [51]. First, the confusion matrix C is calculated. Each value $c_{i,j} \in C$ represents the number of pixels of class i that is predicted as pixels of class j. Let $Y_t^{(i)}$ be the binary map where class i is detected in the label Y_t, such that $Y_t^{(i)}[h,w] = 1 \iff Y_t[h,w] = i$. The confusion matrix can be calculated from the ground truth label Y_t and predicted label $\hat{Y}_t$ as:

$$c_{i,j} = \sum_{h,w} \left(Y_t^{(i)} \odot \hat{Y}_t^{(j)}\right)[h,w] \tag{4.24}$$

The IoU for each class can be found as:

$$\mathrm{IoU}_i\left(Y_t, \hat{Y}_t\right) = \frac{c_{i,i}}{\sum_j c_{i,j} + \sum_j c_{j,i} - c_{i,i}} \tag{4.25}$$

Usually, the mean IoU is used as a single value to evaluate the predictions for all classes. Let the number of classes be n. The mean IoU is then given by:

$$\text{mean IoU}\left(Y_t, \hat{Y}_t\right) = \frac{1}{n} \sum_i \frac{c_{i,i}}{\sum_j c_{i,j} + \sum_j c_{j,i} - c_{i,i}} \tag{4.26}$$

To calculate the (mean) IoU on a whole dataset instead of a single prediction, the confusion matrix is accumulated for each prediction, and then the (mean) IoU is calculated from it.

4.4.2 Temporal Intersection over Union

To measure the temporal stability of consecutive segmentations, we propose the temporal IoU. The idea is as follows. We could use the standard mean IoU $\left(\hat{Y}_{t-1}, \hat{Y}_t\right)$ to measure temporal consistency between two predicted labels. However, this measure would be blind to motion between the two frames, and just wants the labels to be exactly the same. To incorporate the motion, the first frame can be warped along the optical flow, and then the mean IoU $\left(\hat{Y}_{t-1,warped}, \hat{Y}_t\right)$ can be measured. To avoid influence of incorrectly estimated optical flow, the metric should only measure the temporal consistency between pixels of which the optical flow is correct.

To determine the correctness of the optical flow, we use the forward-backward consistency check, proposed by Sundaram et al. [79]. Normally, the backward flow points in the opposite direction as the forward flow:

$$\phi_{t\to t-1}[h, w] = -\phi_{t-1\to t}[h + \phi_{h,t\to t-1}[h, w], y + \phi_{w,t\to t-1}[h, w]] \tag{4.27}$$

If this is not the case at certain positions, the optical flow is locally inconsistent. This can happen because of occlusion occuring between the frames, or because of incorrectly estimated optical flow. To make sure that our evaluation metric does not use incorrect optical flow, these positions must be filtered out. Let ϕ_{bw} and ϕ_{fw} be the left-hand side and the right-hand side respectively of equation (4.27). The binary correctness mask M_t of the optical flow is defined as follows [79]:

$$M_t = \begin{cases} 1, & \|\phi_{bw} - \phi_{fw}\|_2^2 < 0.01 \left(\|\phi_{bw}\|_2^2 + \|\phi_{fw}\|_2^2\right) + 0.5 \\ 0, & \text{otherwise} \end{cases} \tag{4.28}$$

The norm is performed pixel-wise on the values of the optical flow in the h and w direction. Where the forward and backward flow are almost perfectly opposite, the left-hand side of the inequality will become very small. The inequality will therefore hold, so the mask value of this location will be 1, denoting a consistent optical flow. The magic numbers are proposed by the authors [79] and work well. We then define the temporal IoU as:

$$\text{IoU}_{\text{temporal}}\left(\hat{Y}_{t-1}, \hat{Y}_t\right) = \text{mean IoU}^*\left(\hat{Y}_{t-1,warped}, \hat{Y}_t\right) \tag{4.29}$$

where $\hat{Y}_{t-1,warped}$ is $\hat{Y}_{t-1}$ warped along the optical flow according to equation (4.3), and where the confusion matrix C in IoU*, in contrast to equation (4.24), is calculated as:

$$c_{i,j} = \sum_{h,w} \left(M_t \odot \hat{Y}^{(i)}_{t-1,warped} \odot \hat{Y}^{(j)}_t \right) [h, w] \tag{4.30}$$

This metric is perfectly suited to compare the output segmentation consistency of different models. As with the standard IoU, more detailed information can be retrieved from the confusion matrix: for example the $\text{IoU}_{\text{temporal},i}$ specific for a class i, or the false positive or false negative rate. Note however, that the values of this metric are dependent on the quality of the optical flow. As a consequence, metric values produced by implementations with a different optical flow calculation, should not be compared.

For reference, Lai et al. [43] propose a somehow similar metric for temporal stability: the flow warping error between two outputs. This metric is not suited for comparing segmentation labels Y, but can be used to compare the full outputs O instead.

$$E_{warp}(O_{t-1}, O_t) = \frac{1}{\sum_{h,w} M_t} \sum_{h,w} M_t \| O_{t-1,warped} - O_t \|_2^2 \tag{4.31}$$

4.5 Conclusion

This chapter lists the network architectures that will be used in the experiments, the loss functions that will be used to train them, and the evaluation metrics that will assess the quality of their results. The baseline architecture is ENet [63]. All other network architectures are recurrent variations of this ENet. The ENet + appended ConvGRU architecture, based on the work of Nilsson et al [57] is such a variation. It appends a ConvGRU at the end of the ENet, to make a final prediction for the current timestep based on both the proposed prediction by ENet and the final prediction of the previous timestep. Because of its dependency on qualitative dense optical flow at runtime, this network is prone to errors in the optical flow estimation. Thereby it makes it less suitable for real-time applications, since the calculation of optical flow is time-consuming. The variations that will be used during the experiments are the ENetLSTM and the ENetGRU, based on the work by Lai et al. [43]. Instead of appending a recurrent block to the ENet network, they insert it in its center, between the encoder and the decoder part. Small movements of objects in the scene have less influence on the local representations in the center of the network, because the spatial compression is much larger. Therefore, the small movements don't have to be corrected by optical flow.

Every network is trained to minimize a loss function on a certain dataset. Two types of loss functions are used in this thesis: an accuracy loss funtion and a temporal consistency loss function. As accuracy loss function, the cross-entropy loss is used. This loss is widely used in semantic segmentation tasks, and learns the network to produce segmentations that are as close as possible to the ground truth segmentations. The three temporal consistency losses are the change, smoothing and warping loss.

These losses want the network to produce temporally consistent predictions. The change loss is the ultimate temporal consistency loss. It penalizes any difference between consecutive predictions. The smoothing loss is a more relaxed variant. This loss penalizes small displacements in the predictions less than the change loss. After all, scenes are rarely fully static. The displacements, however, may occur in all directions equally likely. Both the change and the smoothing loss are new losses proposed in this thesis. The smoothing loss is not tested in the evaluations because of time shortage. Finally, the warpingloss is the most advanced one of the three. It uses optical flow to determine in which direction the objects in the scene actually move, and penalizes the predictions that didn't displace accordingly. The warpingloss is also proposed by Lai et al. [43]. To train a network, the final loss will be a weighted sum of the cross-entropy loss and one of the temporal consistency losses.

Finally, the combinations of network architectures and loss functions will be evaluated with two metrics: the standard IoU metric and the temporal IoU metric. The standard IoU metric is widely used in semantic segmentation to assess the accuracy of a prediction given its ground truth label. The temporal IoU is a new metric proposed by this thesis, and assesses the temporal consistency between two consecutive predicted labels, using optical flow warping. It is perfectly suited to compare the segmentation consistency of different models, and can be broken down in more detailed information like the temporal IoU for each single class. However, because of its dependence on optical flow estimates, temporal IoU values produced with different optical flow calculation algorithms should not be compared.

Chapter 5

Evaluation

The previous chapter listed multiple network architectures, loss functions and evaluation metrics. This chapter describes how they are tested, and shows the evaluation results. First of all, section 5.1 describes the framework. The framework contains all code and libraries that are used to run the evaluations. Section 5.2 describes the datasets used for testing. Section 5.3 then describes the experiments and their results. Finally, the last section summarizes the conclusions of the experimental evaluations.

5.1 Framework

The framework is the collection of all code and libraries that are used to do the evaluation. It is built in such a way that different architectures, datasets and loss functions can easily be plugged in. It is written in Python, and makes use of the PyTorch library for neural network computing. PyTorch does not only allow GPU training and high-level neural network programming, it is also easily ported to the hardware/software for autonomous cars.

Most research on deep learning works with text or image data. Since video data is less common, no default implementations exist for video-specific building blocks. For example, default dataloaders for images or text exist in PyTorch. However, full video sequences can, in contrast to images or text, not be loaded into memory at once. As they must be loaded frame per frame, the networks, loss functions and metrics must have a state. To be able to reset the states at the correct time, it needs to be clear when an image belongs to the current video sequence or a new one. Therefore, all datasets in our framework load three items per image: the image itself, optionally the corresponding label, and a flag indicating whether the image is a successor of the previous image. When an image is not flagged as a successor, all states will be reset before processing it. This allows to abstract the differences between images, sequential images and video files to other parts of the data. Also, when loading video files, thread safety must be taken care of. Thereby, we want to load multiple videos in parallel to be able to train on minibatches of video data. To

make this possible, custom dataloaders have to be written. A simplified UML class diagram of this custom data loading architecture is shown in figure 5.1, 5.2 and 5.3.

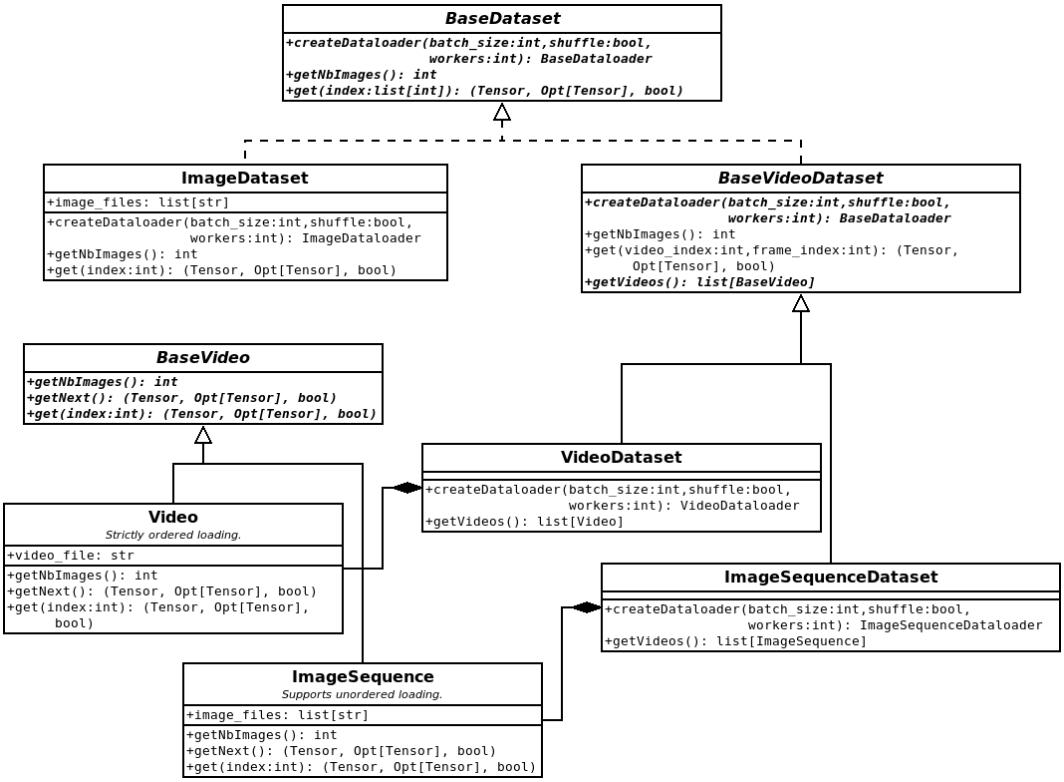

Figure 5.1: Simplified *Dataset* UML class diagram. A dataset keeps all metadata and implements how the data must be loaded. The framework differentiates between image data, image sequence data and video data. An image sequence consists of image files that are temporally related. It can be for example a video of which each frame is stored as an image. It differs from video data, since the individual images can be loaded concurrently. Contrary, videos must be loaded sequentially in the same thread.

Other video-specific building blocks that have no default implementation in PyTorch (yet), are the ConvLSTM and ConvGRU modules. Therefore, we also implemented them ourselves. The implementations are based on the Github repositories of Andrea Palazzi [61] and Jacob Kimmel [39], but were modified and extended to implement a more uniform and easier interface. The ENet implementation is also based on a Github repository from David Silva [73].

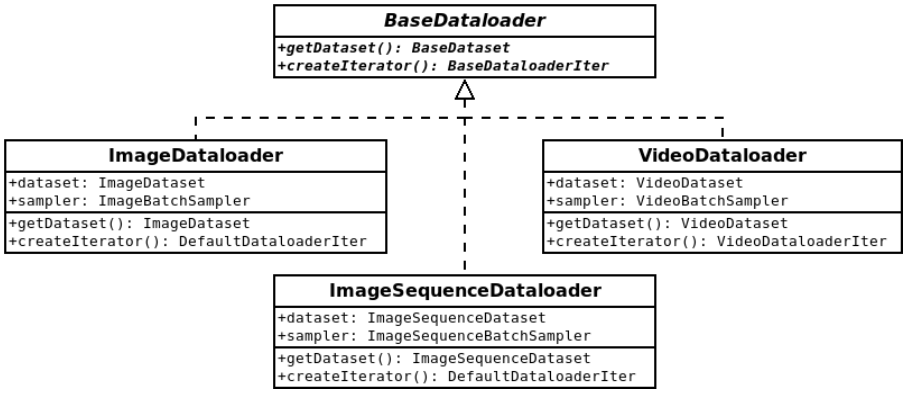

Figure 5.2: Simplified *Dataloader* UML class diagram. A dataloader combines a dataset and a sampler. A sampler determines the order in which the elements of the dataset must be loaded. A dataloader has no state. Different classes are created for each type of data, since they need other types of samplers, and since a *VideoDataloader* must be more thread-safe.

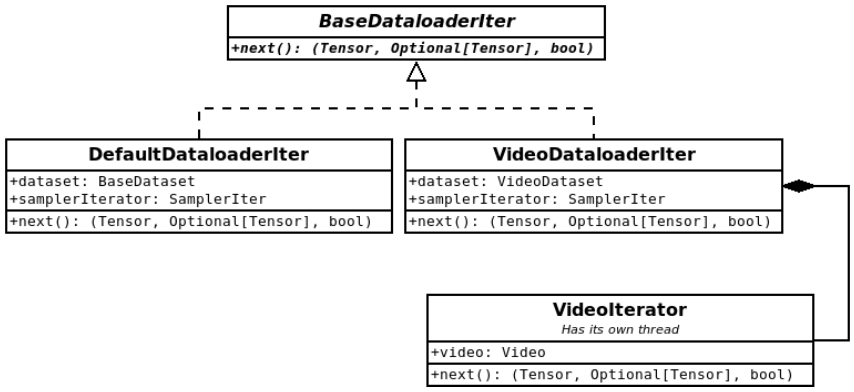

Figure 5.3: Simplified *DataloaderIter* UML class diagram. A dataloader iterator keeps track of which images are already loaded, and which still have to be loaded. Therefore, in contrast to the dataloader, a dataloader iterator does have a state. The advantage of this division of responsabilities, is that multiple iterators over the same dataloader can be made at the same time without interfering with each other.

5.2 Datasets

This thesis depends on datasets that contain video data from street scenes. The two datasets used are Camvid [7][6] and Cityscapes [18]. Since most recurrent networks overfitted on the CamVid dataset, the bigger Cityscapes dataset was used for the final experiments. This section covers the most important properties of both datasets.

CamVid semantic classes			
Fixed objects	Moving objects	Road	Ceiling
Building	Animal*	Road	Sky
Wall*	Pedestrian	Shoulder*	Tunnel*
Tree	Child*	Drivable and markings*	Archway*
Vegetation misc*	Rolling cart / luggage*	Non-drivable*	
Fence	Bicyclist		
Sidewalk	Motorcycle*		
Parking block*	Car		
Column / pole	SUV / pickup truck*		
Traffic cone*	Truck / bus*		
Bridge*	Train*		
Sign / symbol	Misc*		
Misc text*			
Traffic light*			
Other*			

Table 5.1: Table containing all semantic classes in CamVid. The starred classes are not (separately) learned, because they don't appear enough in the dataset.

5.2.1 CamVid dataset

The Cambridge-driving Labeled Video Database or CamVid dataset [7][6] contains image- and video data of street scenes. They are all filmed by a camera mounted on the front of a car. The video data has a resolution of 720x960, is filmed at 30Hz, and shows scenes of urban environments in different light conditions. It contains 701 finegrained pixel-wise semantic labels. They are provided at 1Hz or 15Hz, dependent on the video sequence. The dataset differentiates between 32 semantic classes, of which 12 are used during evaluation. The others are generally not used because they don't appear enough in the dataset. The semantic classes are shown in table 5.1.

5.2.2 Cityscapes dataset

Cityscapes [18] is a much larger dataset. It contains stereo video sequences of street scenes, along with coarse and fine pixel-wise labels, GPS coordinates, temperature measurements and much more. The footage is also filmed by a camera mounted on the front side of the car. The video data has a resolution of 1024x2048, and is filmed at 30Hz in good light conditions. It contains 5000 finegrained pixel-wise labels. The dataset differentiates between 30 semantic classes, of which 19 are used during evaluation. The others are generally not used because they don't appear enough in the dataset. The semantic classes are shown in table 5.2. In the experiments, all Cityscapes images are resized to a resolution of 256x512 to speed-up the training and to allow real-time evaluation.

Cityscapes semantic classes			
Fixed objects	Moving objects	Environment	Other
Rail track*	Person	Road	Sky
Building	Rider	Sidewalk	Tunnel*
Wall	Car	Parking*	Ground*
Fence	Truck	Vegetation	Dynamic*
Guard rail*	Bus	Terrain	Static*
Bridge*	On rails		
Pole	Motorcycle		
Pole group*	Bicycle		
Traffic sign	Caravan*		
Traffic light	Trailer*		

Table 5.2: Table containing all semantic classes in Cityscapes. The starred classes are not (separately) learned, because they don't appear enough in the dataset.

5.3 Results

This section describes the results of the architecture variations and loss functions proposed in chapter 4.2 and 4.3, and compares them using the metrics from section 4.4. As the baseline case, we will use a standard ENet [63] trained with a cross-entropy loss. The weights of this ENet will be used to initialize the other ENet-based architectures. However, This section does not evaluate the ENet + appended GRU architecture [57] and the smoothing loss. As the ENet + appended GRU is highly dependent of qualitative optical flow, inaccuracies in the optical flow influence its prediction in a negative way. Thereby, its inference is very slow and not suitable for real-time applications. The smoothing loss is not evaluated because of time shortage.

All results in this section will be evaluated on video sequences of 10 frames, where the 9th frame has a ground truth label available. The mean IoU metric, which compares a predicted label to the ground truth label, will therefore only be calculated every 9th frame. The mean temporal IoU, assessing the temporal consistency between the previous and current prediction, can be calculated for every frame in range 2 to 10. It will be interesting to show its evolution during the video sequence. However, the Cityscapes dataset labels some pixel regions as *void* or *don't care*. These regions should not be included in the loss function and accuracy metric. After all, a lot of consistency might be gained in those regions. An example is the hood of the ego-vehicle, which is visible in every frame. The predictions of this *don't care* region are often noisy. To allow a fair comparison, a second version of the mean temporal IoU will be measured, where this region is ignored. To know exactly which regions should be ignored, the ground truth label is needed. Therefore, this metric will only be calculated at every 9th frame too.

5.3.1 ENet

The ENet [63] is trained on Cityscapes. We use the *Adam* optimizer [40], with learning rate 0.0005 and weight decay 0.0002. After 100 timesteps, the learning rate is reduced with a factor 0.1. The batch size is 30. The training statistics can be found in appendix A.1. The checkpoint taken at epoch 103 is used throughout this thesis as the baseline ENet network. All other ENet-based architectures will use the pre-trained weights of this network. Therefore, their training statistics will always start at epoch 104.

Table 5.3 shows the evaluation metrics of this ENet. The mean IoU achieved using these settings, is 48.85%. The mean temporal IoU ignoring the *don't care* regions is 67.15%. The average evolution of the mean temporal IoU over the sequence of 10 frames, is shown in figure 5.4. As can be expected, this temporal consistency is constant, as ENet has no recurrent connections to access information of previous frames.

	Metrics (%)	
Arch.	mean IoU	mean temp. IoU ign.
ENet	48.85	67.15

Table 5.3: ENet evaluation results on the Cityscapes dataset with reduced image resolution of 256x512.

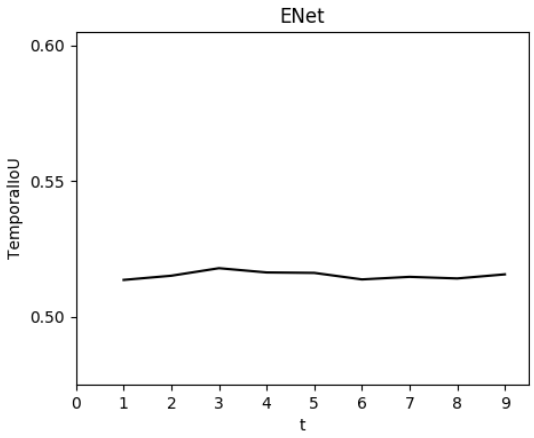

Figure 5.4: This figure shows the average evolution of the mean temporal IoU during video sequences of 10 frames.

5.3.2 ENetGRU / EnetLSTM

A recurrent layer inserted between the encoder and the decoder part of ENet, should enable the network to learn temporal consistency. Here, we test the effects of both a ConvGRU and a ConvLSTM layer, resulting in the ENetGRU and ENetLSTM. For each of them, multiple configurations are tested.

The first configuration difference that is tested is the kernel size. A smaller kernel size is expected to result in faster calculations, while a larger kernel size should have better accuracy. Another configuration difference that is tested, is the combination of the recurrent layer with the residual connection. The result of the recurrent layer could be added or appended to the residual connection, as visualized in figure 4.6b. Here too, the addition is expected to be more time efficient, while the concatenation is expected to give better accuracy. In case of the concatenation, the network itself learns how to combine the two information streams. In the results, these variations will be named as follows: ENet{LSTM/GRU}-ksize{1/3}-{concat/summate}. For example, ENetGRU-ksize1-concat refers to the variation of the ENetGRU with kernel size 1 and with concatenation of the residual connection.

Each network configuration is tested with the change- and warping loss function. Their results will be compared with the standard ENet results, but for a fair comparison, they will also be compared to the results of the same network without a temporal loss function. After all, the proposed networks add at least 1 layer to the standard ENet, which gives it more expressive power. An increasement in accuracy or consistency may therefore be solely due to this extra expressiveness and not due to the added temporal loss function. In this tests: the cross-entropy loss and change loss are combined according to equation (4.6) with weights 0.8 and 0.2 respectively, and the cross-entropy loss and warping loss are combined with weights 0.99 and 0.01 respectively. These weights are based on the scale of both losses, are empirically tested and have good results. The optical flow estimation is calculated using the Lucas-Kanade algorithm [52]. The training plots for all trained networks can be found in appendix A. All recurrent networks are trained with 1 backpropagation-through-time step. This means that the loss propagates back through the recurrent connection up to the input of the previous timestep. We use the *Adam* optimizer [40], with learning rate 0.001 and weight decay 0.0002. After 50 timesteps, the learning rate is reduced with a factor 0.31. The batch size is 7. The training statistics can be found in appendices A.2 and A.3.

ConvLSTM

The accuracies of the ENetLSTM variations, measured by the mean IoU, are listed in table 5.4. All networks, with or without a teporal loss, have a mean IoU that is larger than the standard ENet. So the recurrent connections significantly improve the accuracy. The addition of a temporal loss function generally improves the accuracy even slightly more. The best accuracy is achieved by ENetLSTM-ksize3-summate, 3.07% above the accuracy of the baseline ENet network.

The temporal consistency, as listed in table 5.5, improves too when a temporal loss is applied. This table shows that the best result is achieved by the ENetLSTM-ksize3-concat architecture. Compared to the standard ENet, this network with change loss improves with 3.02%, and with warping loss it improves with 4.27% on the mean temporal IoU ign. metric.

Figure 5.5 shows the evolution of the mean temporal IoU metric for the four architecture variations without a temporal loss. The evolution is averaged over all 10-frame sequences in the Cityscapes dataset. This figure shows that, even without temporal loss, the temporal consistency increases as more frames of the video are processed. Figure 5.6 shows that this effect is even more apparent when a temporal loss is used during training. Both the change loss and the warping loss clearly improve the temporal consistency. Both losses have a similar consistency

	Mean IoU (%)		
Arch. / Temporal loss	None	Change	Warp
ENet	48.85	/	/
ENetLSTM ks1 conc	50.40	50.23	**50.47**
ENetLSTM ks1 summ	50.11	50.58	**50.95**
ENetLSTM ks3 conc	50.28	**51.46**	50.93
ENetLSTM ks3 summ	50.51	**51.92**	50.75

Table 5.4: ENetLSTM: mean IoU metric. The ks{1/3} indicates the kernel size of the ConvLSTM, and the conc/summ indicates whether the result of the ConvLSTM is concatenated or appended to the residual connection. The extension with the ConvLSTM clearly improves the segmentation accuracy, compared to the baseline ENet network. An additional temporal loss function slightly improves the accuracy even more.

ConvGRU

The accuracies of the ENetGRU variations, measured by the mean IoU, are listed in table 5.6. Here, all networks trained by a temporal loss have a larger mean IoU than the standard ENet. The ENetGRU-ksize3-concat architecture has the largest mean IoU, 1.91% better than the standard ENet. The networks trained by a temporal loss also have a larger mean IoU than the networks trained without temporal loss. So here, the temporal losses clearly improve the accuracy.

The temporal consistency, as listed in table 5.7, generally improves when training with a temporal loss. However the improvements are less convincing than those of the ENetLSTM. The table shows that the best result is again achieved by the ENetGRU-ksize3-concat architecture. Compared to the standard ENet, this network without temporal loss improves with 3.94%, with change loss it improves with 7.99%, and with warping loss it improves with 9.43% on the mean temporal IoU ign. metric.

	Mean temporal IoU ign. (%)		
Arch. / Temporal loss	None	Change	Warp
ENet	67.15	/	/
ENetLSTM ks1 conc	65.13	67.42	**68.70**
ENetLSTM ks1 summ	64.97	66.99	**68.42**
ENetLSTM ks3 conc	66.07	70.17	**71.42**
ENetLSTM ks3 summ	65.46	68.06	**68.53**

Table 5.5: ENetLSTM: mean temporal IoU metric with *don't care* regions ignored. The ks{1/3} indicates the kernel size of the ConvLSTM, and the conc/summ indicates whether the result of the ConvLSTM is respectively concatenated or appended to the residual connection. Clearly, the warping loss improves the temporal consistency of all architecture variations. The ENetLSTM-ksize3-concat has good results with both temporal loss functions.

Figure 5.7 shows the evolution of the mean temporal IoU metric for the four architecture variations without a temporal loss. The evolution is averaged over all 10-frame sequences in the Cityscapes dataset. This figure shows that, even without temporal loss, the temporal consistency increases as more frames of the video are processed. Figure 5.8 shows that this effect is even more apparent when a temporal loss is used during training. Both the change loss and the warping loss clearly improve the temporal consistency. The warping loss has the best results. In figures 5.9 and 5.10, qualitative results of the standard ENet and the ENetGRU-ksize3-concat trained with warping loss are visualized.

	Mean IoU (%)		
Arch. / Temporal loss	None	Change	Warp
ENet	48.85	/	/
ENetGRU ks1 conc	49.24	49.30	**49.97**
ENetGRU ks1 summ	48.76	48.99	**50.09**
ENetGRU ks3 conc	50.28	50.46	**50.76**
ENetGRU ks3 summ	49.30	50.26	**50.51**

Table 5.6: ENetGRU: mean IoU metric. The ks{1/3} indicates the kernel size of the ConvGRU, and the conc/summ indicates whether the result of the ConvGRU is concatenated or appended to the residual connection. All networks trained with a temporal loss have a higher mean IoU than the standard ENet. The warping loss delivers the best results.

	Mean temporal IoU ign. (%)		
Arch. / Temporal loss	None	Change	Warp
ENet	67.15	/	/
ENetGRU ks1 conc	69.64	**73.27**	69.39
ENetGRU ks1 summ	67.66	65.70	**72.49**
ENetGRU ks3 conc	72.16	75.14	**76.58**
ENetGRU ks3 summ	64.81	**74.20**	68.15

Table 5.7: ENetGRU: mean temporal IoU metric with *don't care* regions ignored. The ks{1/3} indicates the kernel size of the ConvGRU, and the conc/summ indicates whether the result of the ConvGRU is respectively concatenated or appended to the residual connection. The temporal losses generally improve the temporal consistency, however with varying success.

5.3.3 Execution speed

For every architecture, we measured the execution speed. The results are listed in table 5.8. They are measured on a GeForce GTX 1080 Ti graphics card, on image sequences with resolution 256x512. The recurrent architectures are a little bit slower than the standard ENet architecture, since they have more layers. Counter-intuitively, the ConvGRU is slower than the ConvLSTM, although it is more compact. However, the differences are almost negligible: the slowest architecture is 2.9% slower than the standard ENet. To be considered real-time, a network should achieve a framerate of at least 10 frames per second (fps). All tested networks clearly meet this requirement. They even allow real-time processing of higher resolution image sequences or real-time processing of multiple image sequences in parallel.

Architecture	Execution speed (fps)
ENet	86.3
ENetLSTM ks1 conc	85.3
ENetLSTM ks1 summ	85.4
ENetLSTM ks3 conc	84.5
ENetLSTM ks3 summ	85.5
ENetGRU ks1 conc	83.8
ENetGRU ks1 summ	84.6
ENetGRU ks3 conc	83.8
ENetGRU ks3 summ	84.6

Table 5.8: Execution speeds of the different architectures on a GeForce GTX 1080 Ti graphics card on images with resolution 256x512.

5.4 Conclusion

The experiments show that recurrent network architectures are well suited to improve the temporal consistency of semantic video segmentations. We tested multiple recurrent variations of the ENet architecture. These include variations over the type of recurrent block (ConvLSTM or ConvGRU), the kernel size of the recurrent block (1 or 3), and the way it is inserted in the ENet network (concatenated to or summated with the residual connection). Each network variation is tested with three different loss functions: the classic cross-entropy loss, the combination of the cross-entropy loss with the change loss, and the combination of the cross-entropy loss with the warping loss. Every combination is evaluated on accuracy (mean IoU) and on temporal consistency (mean temporal IoU).

The ConvLSTM based network architectures significantly improve the accuracy with respect to the ENet baseline. The temporal consistency also clearly improves when a temporal loss function like the warping loss is used during training. The ConvLSTM based architecture with the best results, is the ENetLSTM-ksize3-concat trained with the warping loss. It improves the accuracy with 2.08% and the temporal consistency with 4.27%, compared to the standard ENet. The ConvGRU based network architectures also clearly improve the accuracy. Their temporal consistency improves with varying success. The best ConvGRU based architecture is the ENetGRU-ksize3-concat trained with the warping loss. It improves the accuracy with 1.91% and the temporal consistency with 9.43%, compared to the standard ENet.

The experiments also show that the temporal loss functions influence the accuracy and temporal consistency in a positive way. The improvements are not solely due to the extra expressiveness of the extra recurrent layers. Indeed, the network architectures trained with temporal losses generally score higher on both metrics, compared to the same network architectures trained without temporal losses.

The plots of the temporal IoU metric visualize how the recurrent networks build up an internal memory of the scene. The temporal consistency clearly increases during the first three frames, and then remains constant during the next frames. This effect even occurs when no temporal loss is used. So even without a temporal loss, the network learns to use the information of previous frames. However, a temporal loss improves this effect.

Finally, to be applicable in real-time contexts, the networks should achieve at least a framerate of 10 frames per second. The addition of the recurrent block slows down the standard ENet with only 2.9%. Therefore, they all are able to process videos with a resolution of 256x512 at 83.8 frames per second, on a GeForce GTX 1080 Ti. This leaves enough headroom to process videos at higher resolution or to process multiple videos in parallel.

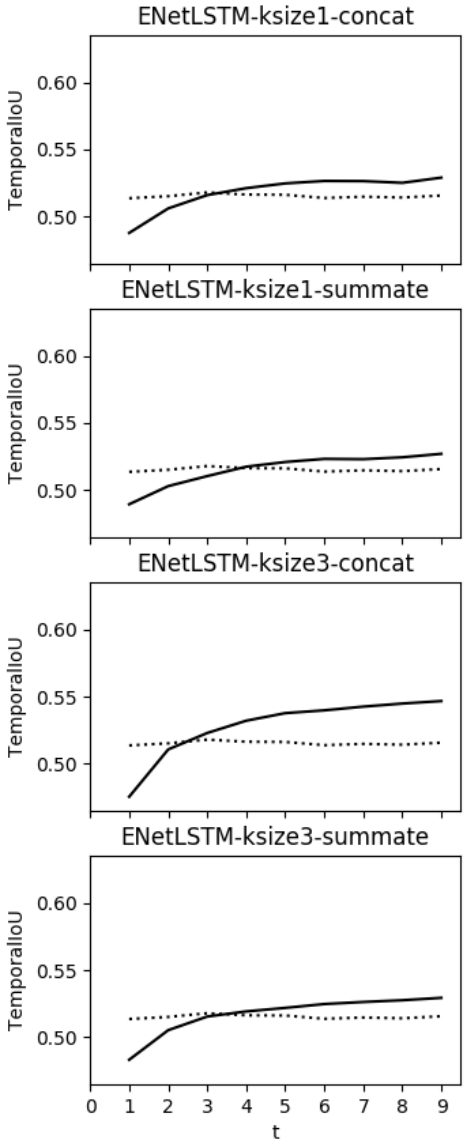

Figure 5.5: This figure shows the average evolution of the mean temporal IoU during video sequence of 10 frames. The ENetLSTM network is trained without a temporal loss (full line). Even without a temporal loss, the temporal consistency improves over time. For comparison: the average evolution of the mean temporal IoU of the standard ENet is visualized too (dotted line).

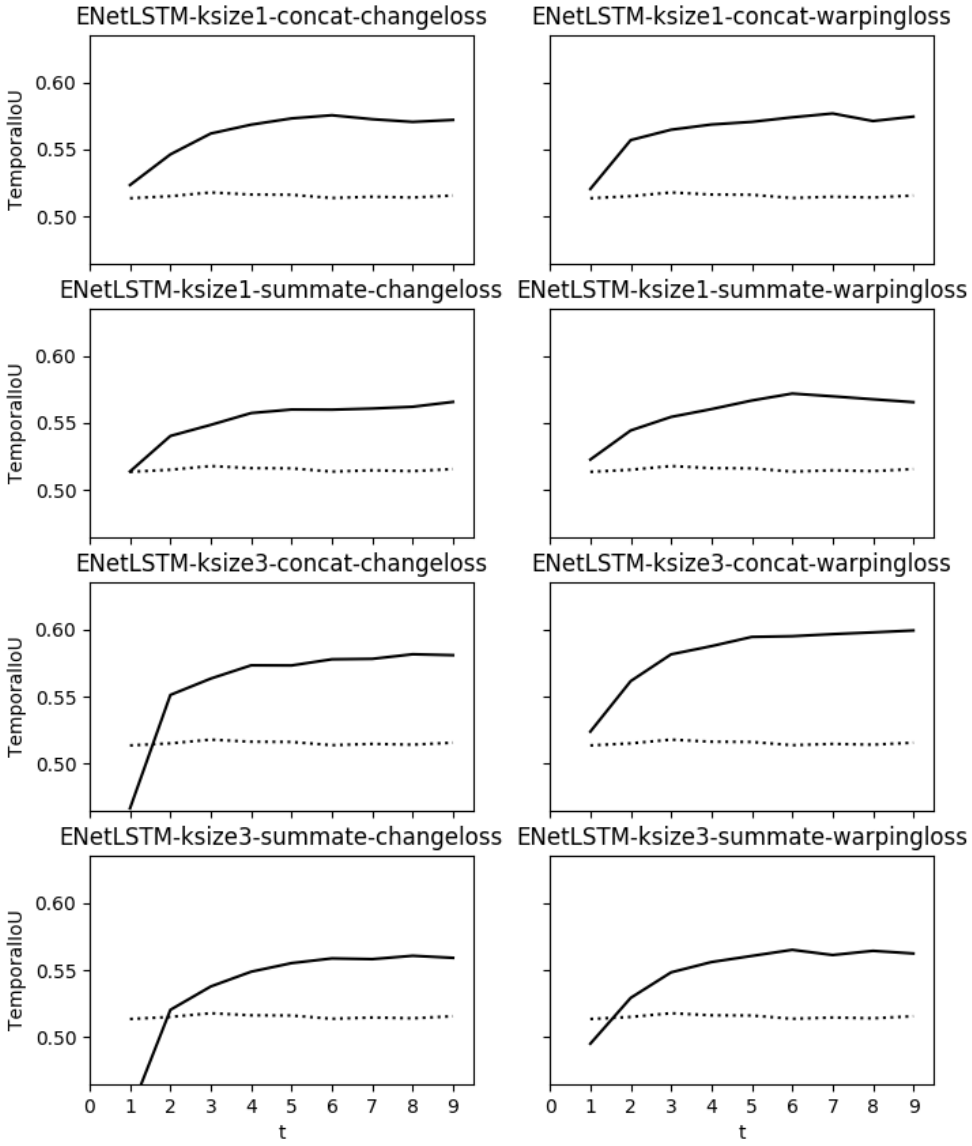

Figure 5.6: This figure shows the average evolution of the mean temporal IoU during video sequence of 10 frames. The ENetLSTM network is trained with the change- or warping loss (full line). These losses clearly exploit the recurrent information, resulting in better consistency. The warping loss has the best results. For comparison: the average evolution of the mean temporal IoU of the standard ENet is visualized too (dotted line).

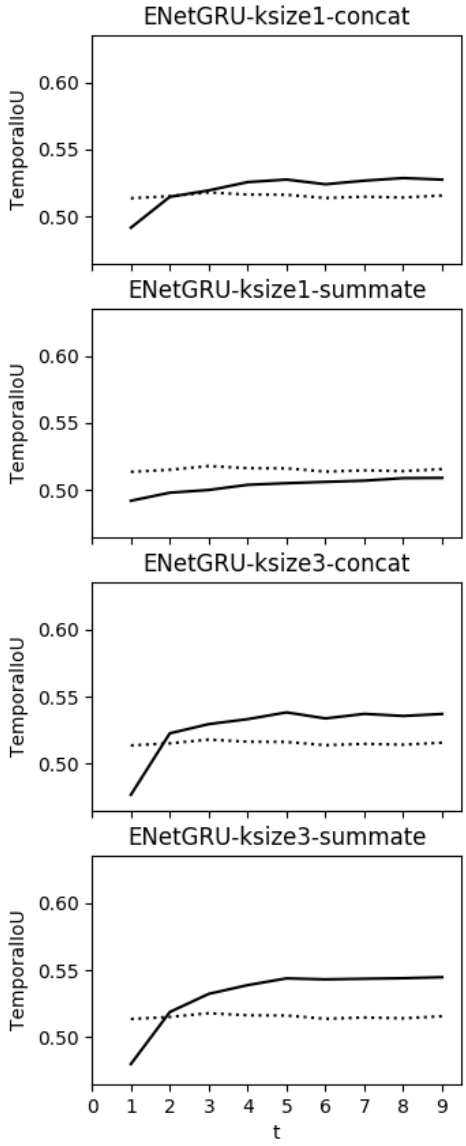

Figure 5.7: This figure shows the average evolution of the mean temporal IoU during video sequence of 10 frames. The ENetGRU network is trained without a temporal loss (full line). Just like with the ENetLSTM, the temporal consistency improves over time, even without temporal loss. For comparison: the average evolution of the mean temporal IoU of the standard ENet is visualized too (dotted line).

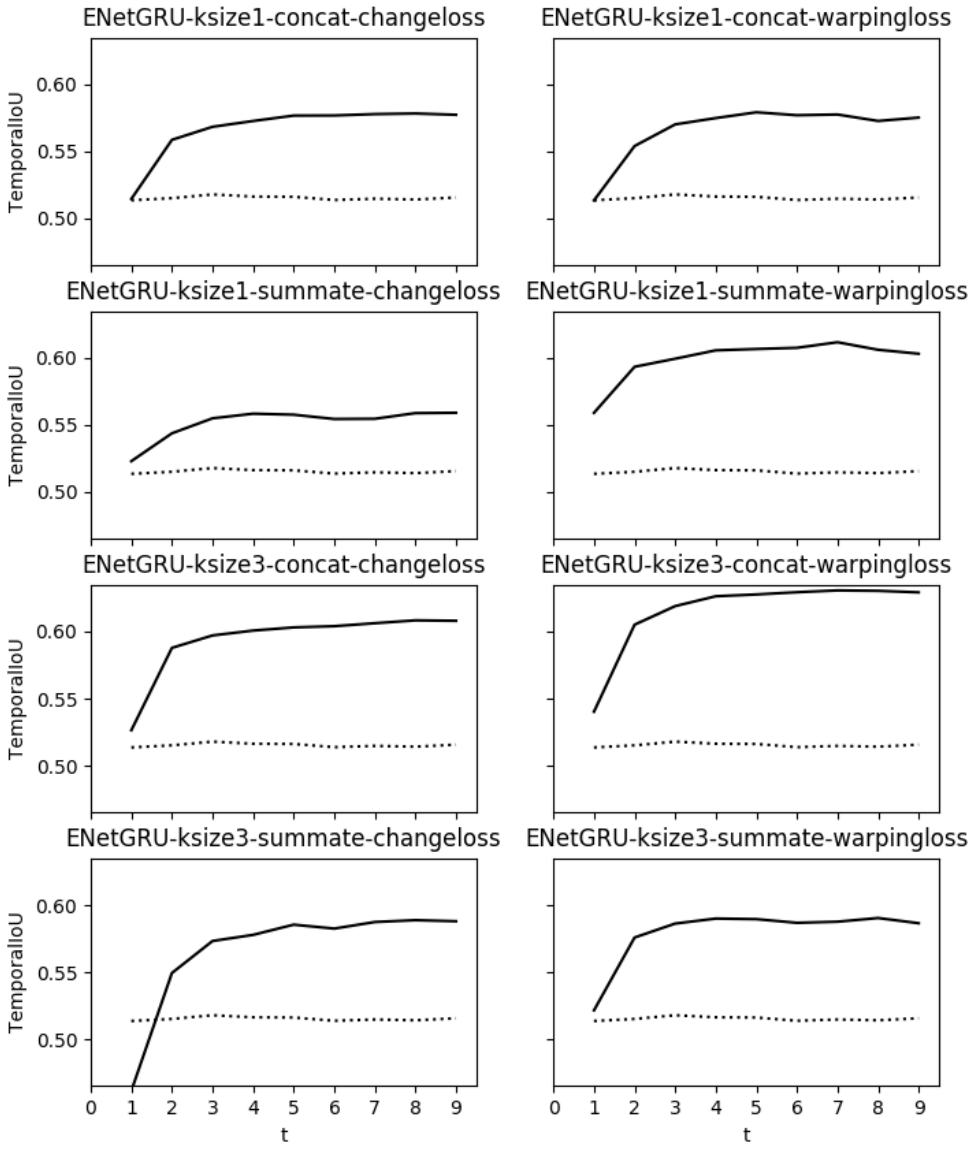

Figure 5.8: This figure shows the average evolution of the mean temporal IoU during video sequence of 10 frames. The ENetGRU network is trained with the change- or warping loss (full line). These losses clearly exploit the recurrent information, resulting in better consistency. The warping loss has the best results. For comparison: the average evolution of the mean temporal IoU of the standard ENet is visualized too (dotted line).

Figure 5.9: Predictions of ENet [63] (left) and ENetGRU-ksize3-concat trained with warping loss (right) on a video sequence from the Cityscapes [18] dataset. The ENetGRU has a more consistent detection of the bicycle than the ENet.

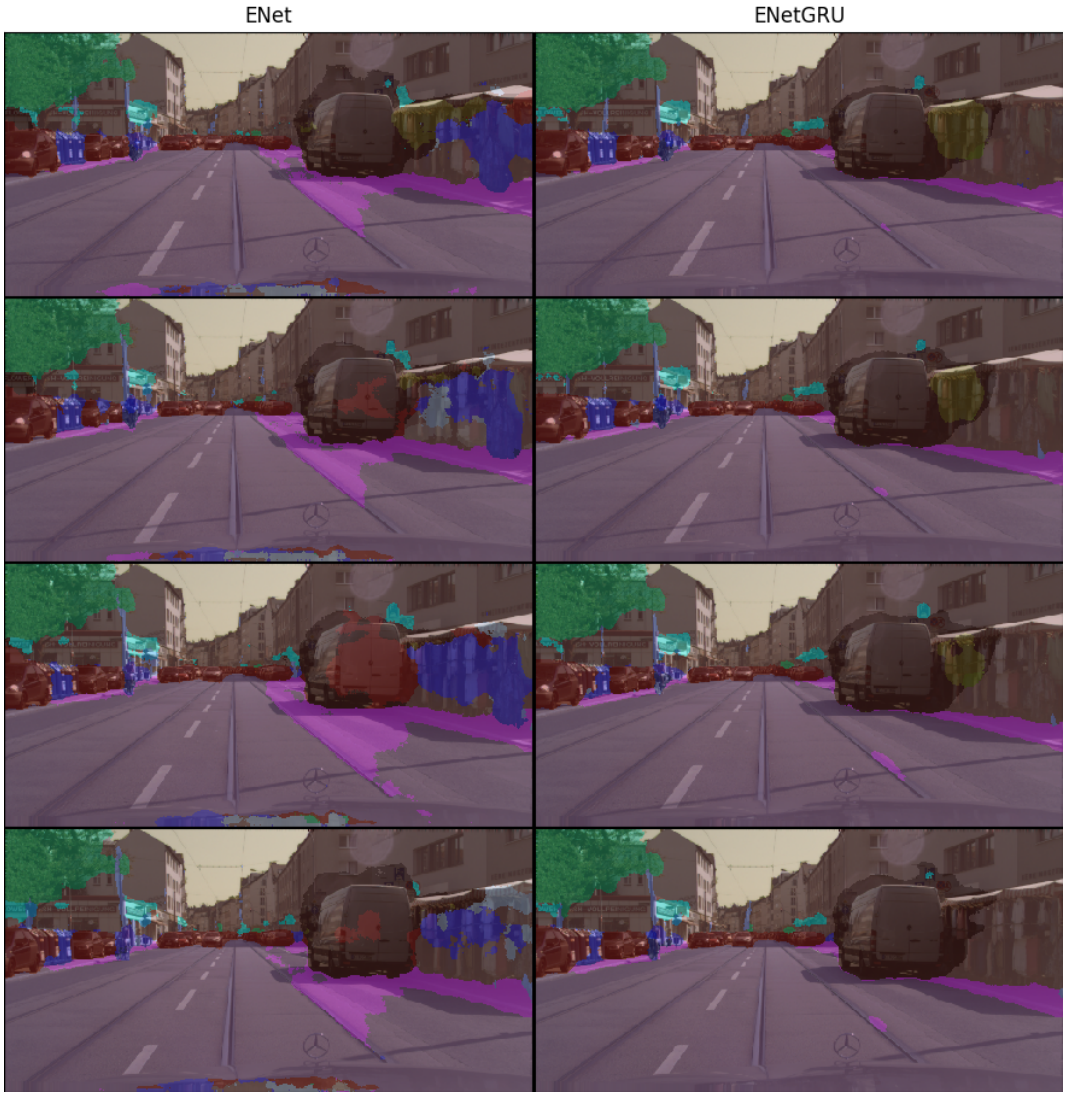

Figure 5.10: Predictions of ENet [63] (left) and ENetGRU-ksize3-concat trained with warping loss (right) on a video sequence from the Cityscapes [18] dataset. The ENetGRU has a more consistent detection of the van and the market stall at the right side of the image.

Chapter 6

Conclusion

6.1 Conclusions

This thesis explores the use of recurrent neural networks to enhance the temporal consistency in real-time semantic video segmentation. The most important manifestation of temporal inconsistency is flickering, when objects alternately are and are not detected. By introducing recurrent connections into the network architecture, the network is able to build up a memory of the previous frames. This memory positively effects the temporal consistency, as the network can additionally use information of previous timesteps to segment the frame at the current timestep. To achieve this goal, this thesis focuses on four main aspects.

The first aspect is the exploration of recurrent network architectures for video processing. As feed-forward networks are not able to learn temporal relations between consecutive inputs, recurrent networks are needed. This thesis does not propose new network architecture ideas, but instead experiments with existing ones. The architecture that is used in the majority of the experiments, is ENet [63] with an inserted recurrent layer. Of this architecture, multiple hyperparameter variations are tested: the type of recurrent layer (ConvLSTM or ConvGRU), its kernel size and its implantation in the feed-forward segmentation network. The experiments show that these recurrent network variations effectively learn temporal relations. The architectures that provides the best results, are the ENetLSTM and ENetGRU architectures trained with warping loss, where the convolutional block has kernel size 3 and is concatenated to the residual connection according to figure 4.6a. The ENetLSTM variant improves the accuracy with 2.08% and the temporal consistency with 4.27%, while the ENetGRU variant improves the accuracy with 1.91% and the temporal consistency with 9.43%, compared to the standard ENet.

The second aspect is the search for appropriate loss functions. The experiments show that the recurrent networks learn temporal consistency, even without a temporal loss function. However, adding a temporal loss function improves the consistency even more. This thesis describes three loss functions for temporal consistency, of which two are not mentioned in literature before. All of them should be used on top of the standard cross-entropy loss for semantic segmentation. The change loss

is the most trivial one. It encourages the network to produce exactly the same segmentations for the current input as for the previous input. The smoothing loss is a more relaxed version of the change loss. It accepts movements in the predictions in any direction as long as they are small. The warping loss is the most sophisticated of the three. It uses optical flow in the input to determine the exact movements that occur in the scene, and only accepts these movements in the predictions. The warping loss was proposed by Lai et al. [43]. In this thesis, only the change- and warping loss are tested. They improve both the accuracy and the temporal consistency of the produced segmentations. The warping loss generally gives best results.

The third aspect is the choice of evaluation metrics. As no standard metric exists for assessing the temporal consistency of consecutive segmentations, this thesis proposes the temporal Intersection over Union metric (temporal IoU). The standard IoU metric is generally used in image segmentation to assess the accuracy of a predicted label, given the ground truth label. The temporal IoU uses the same technique to compare two consecutively predicted labels, but includes optical flow warping to match the two predictions given the motion in the scene. It only takes into account the parts of which the optical flow is forward-backward consistent. This metric is very well suited to visualize the evolution of the temporal consistency on video segmentations. However, to compare the temporal consistency of our recurrent networks with the baseline ENet network, we use a modified version of the temporal IoU. After all, a lot of consistency might be gained in regions that have a "void" ground truth label. But since the labels of these regions don't care, the consistency of these regions should also not be measured for a fair comparison. When these "void" segmentations are ignored, this metric is very well suited to compare the temporal consistency of different networks on a full dataset. One sidenote however, is that the value of this metric depends on the quality of the optical flow. Values of this metric with different optical flow implementations should therefore not be compared.

The final aspect is the creation of a framework that allows to test the previous aspects. As semantic video segmentation is not yet practiced a lot in the PyTorch environment, no standard implementations exist for video dataloaders and convolutional recurrent layers. A lot of time went into the creation of this framework. It is publicly available at https://github.com/JohannesVerherstraeten/semantic-video-segmentation.

The goal of this thesis was to enhance the temporal consistency of real-time semantic segmentations. As the results show, including recurrent layers in a neural network architecture effectively enhances the temporal consistency. Additionally applying a consistency loss during training enhances the temporal consistency even more, and also positively affects the accuracy. The addition of a single recurrent block did not have a large impact on the execution time: the networks with recurrent block are only 2.9% slower than without the recurrent block. Therefore, real-time application is no problem.

6.2 Further research

First of all, for further research, the experiments that are conducted in this thesis could be extended. For instance, the change loss function is not yet tested in depth. Also, other weights for the loss functions could be tested, to find a more optimal balance between the different losses. Thereby, all experiments in this thesis have been evaluated on sequences of 10 frames. The effect of the recurrent networks on longer video sequences should be investigated too.

Secondly, this thesis only uses small recurrent variations on the ENet architecture [63]. More powerful recurrent architectures may improve the accuracy and temporal consistency even more. For example, one could place multiple recurrent blocks in sequence, to gather more temporal information. Also, recurrent blocks could be placed in other stages of the network too. Recurrent blocks in later stages of the network may provide more consistency in the predictions, but might be more prone to mistakes because of motion in the scene. Recurrent networks in earlier stages may provide spatio-temporal features to the rest of the network that might be useful for classifiction and that feed-forward have not access to. For example, if the network is in doubt whether an object is a pole or a pedestrian, it can use information of its previous position to determine if it has moved in the scene or not. Such spatio-temporal features may even be learned in the networks used in the experiments of this thesis, but this is not explicitely investigated. A disadvantage however of more complex recurrent networks, is that their training will take more time. After all, the number of paths that the loss will pass during backpropagation through time, increases exponentially with the number of recurrent connections.

A possible research direction that builds further on this spatio-temporal feature extraction, is intra-class differentiation between dynamic and static instances. For example, using spatio-temporal features, a recurrent network should be able to learn to differentiate between moving or static pedestrians or cars. This way, the neural networks would gain more understanding of the environment than currently possible using the standard image-based techniques.

Another possible research direction is to extend the networks used in this thesis to a two-stream architecture. As mentioned in section 2.4, a two-stream network architecture consists of an object recognition stream and a motion detection stream. These streams support each other, and the final detection is based on information of both streams. The networks trained in this thesis learned temporal consistency between frames, but don't have the capacity to recognize motion. Such two-stream networks however, do have that capacity. Being able to detect the motion of different objects in a scene should even more allow to predict the segmentations of the next frames, which in turn enhances the segmentation consistency and improves the understanding of the scene.

Appendices

Appendix A

Training Plots

This appendix holds the training plots associated with the networks from the results section. The networks trained without temporal loss, have one set of plots: the training and validation losses and IoU. The networks trained with temporal loss, have two sets of plots. The second set contains the decomposition of the training and validation loss into the cross-entropy loss and the temporal loss.

A.1 ENet architecture

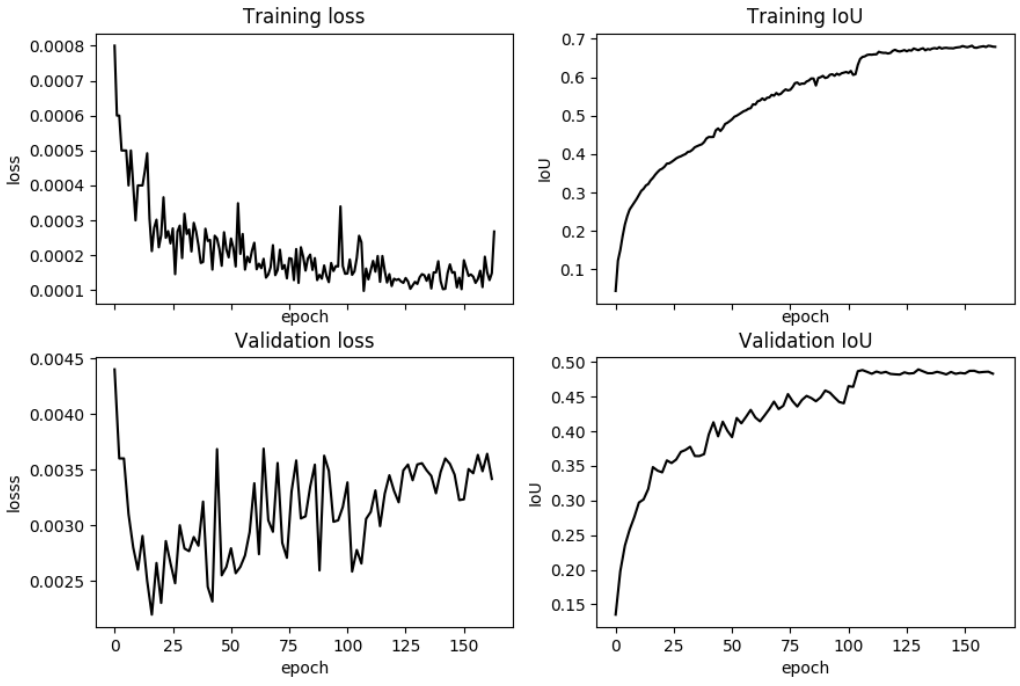

Figure A.1: Training plot for the ENet network.

A.2 ENetLSTM architectures

A.2.1 ENetLSTM-ksize1-concat

No temporal loss

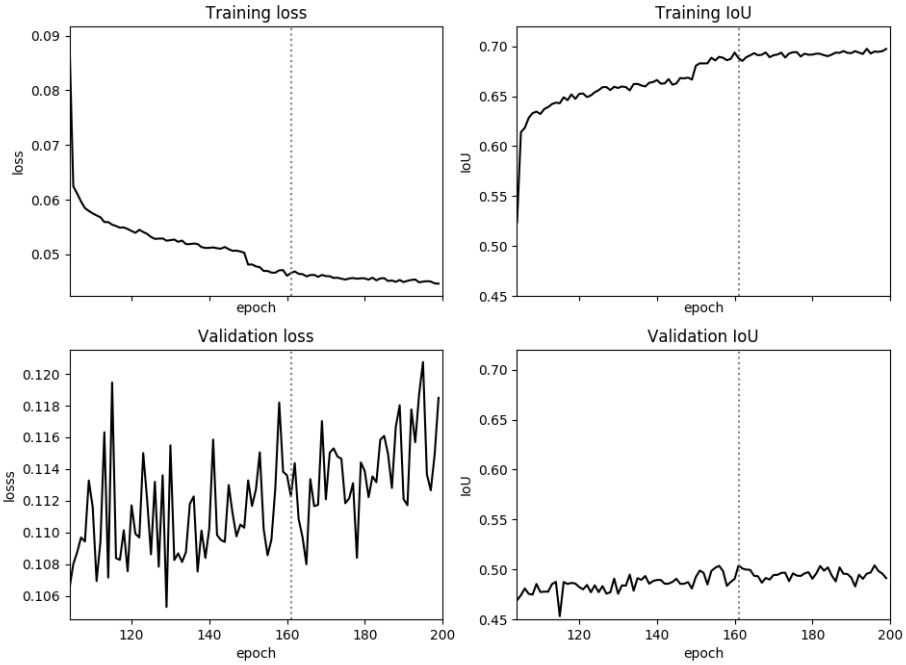

Figure A.2: Training plots for ENetLSTM-ksize1-concat.

Change loss

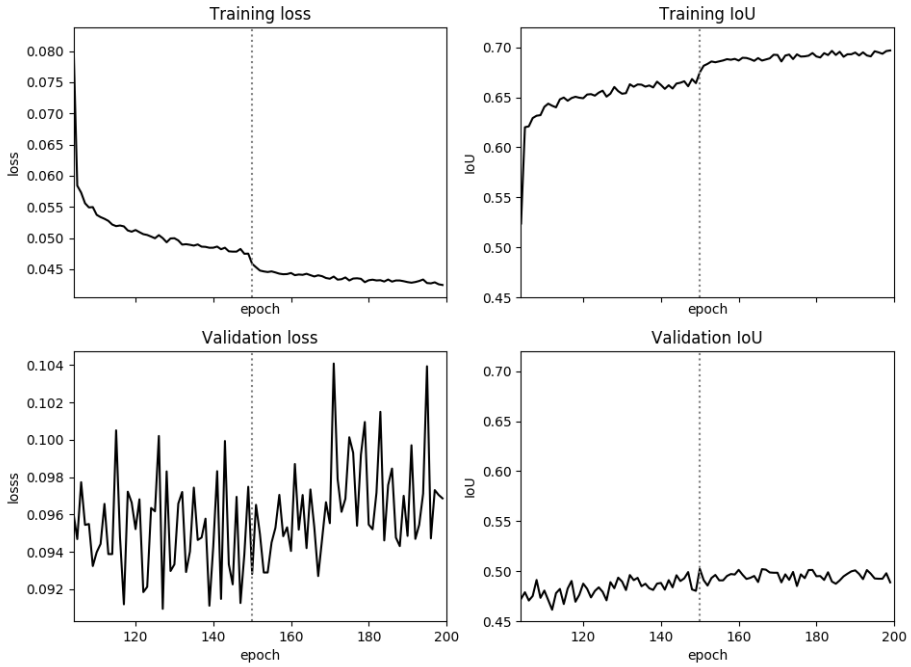

Figure A.3: Training plots for ENetLSTM-ksize1-concat with change loss.

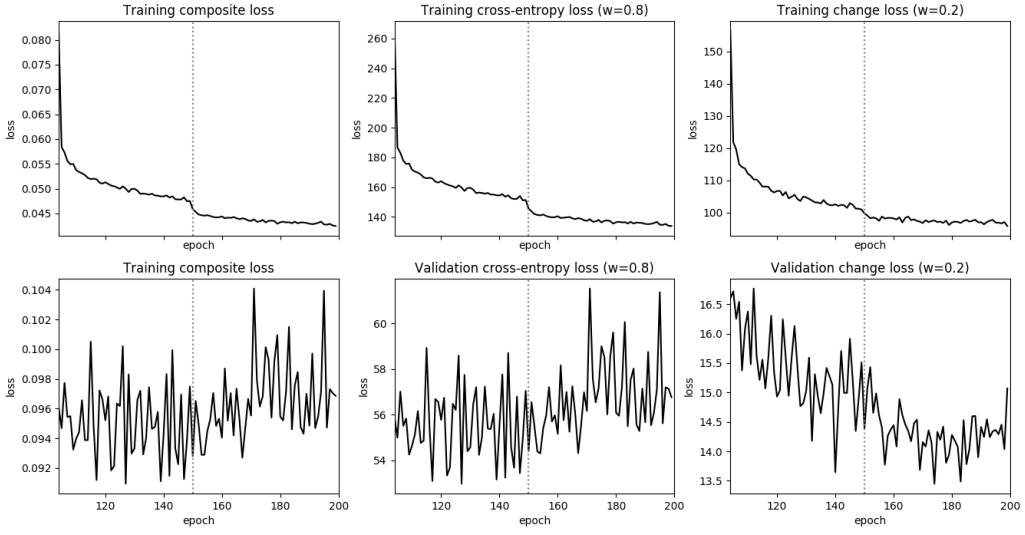

Figure A.4: The decomposition of the losses from figure A.3 into the cross-entropy and change loss. The cross-entropy loss has weight 0.8, and the change loss has weight 0.2.

Warping loss

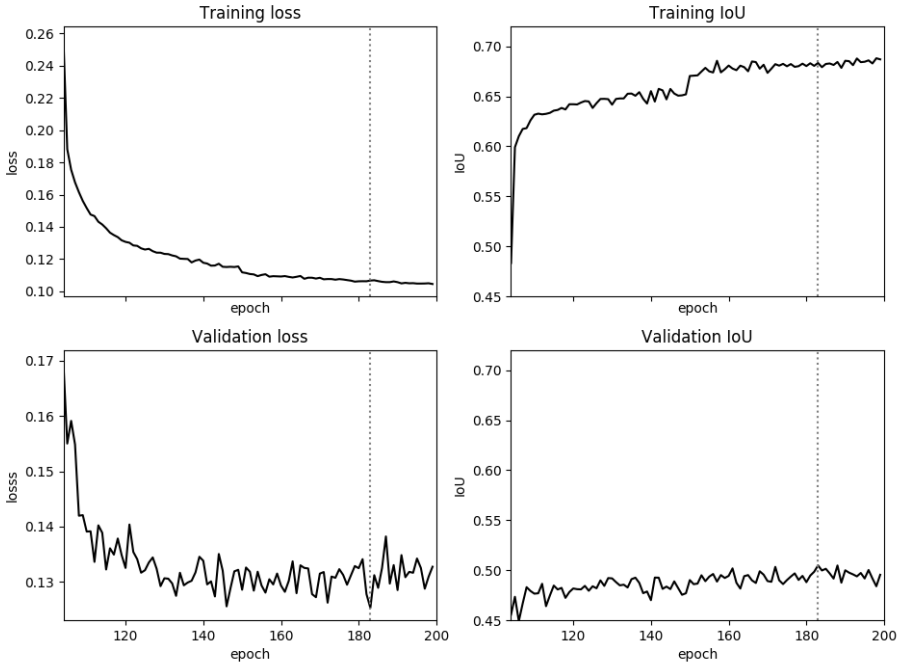

Figure A.5: Training plots for ENetLSTM-ksize1-concat with warping loss.

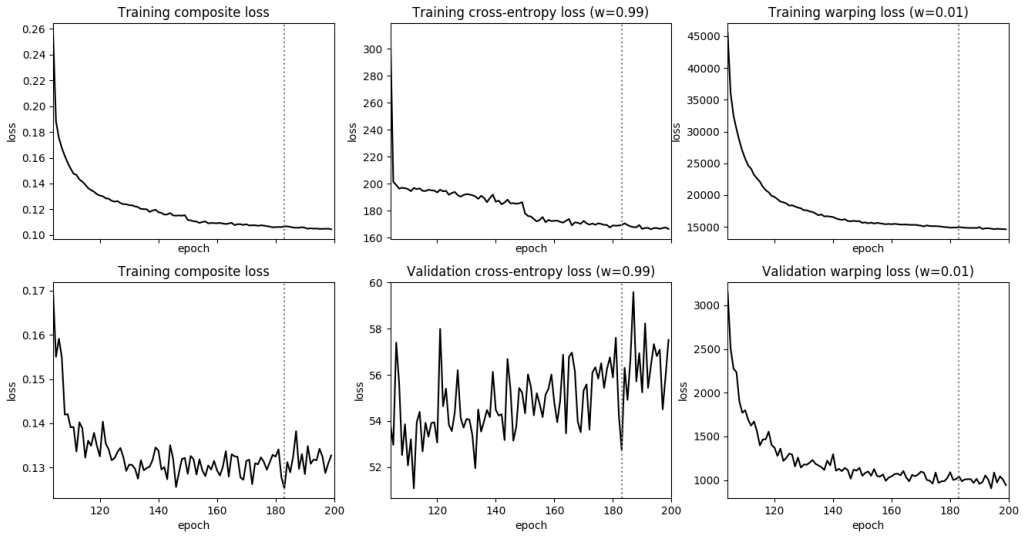

Figure A.6: The decomposition of the losses from figure A.5 into the cross-entropy and warping loss. The cross-entropy loss has weight 0.99, and the warping loss has weight 0.01.

A.2.2 ENetLSTM ksize1 summate

No temporal loss

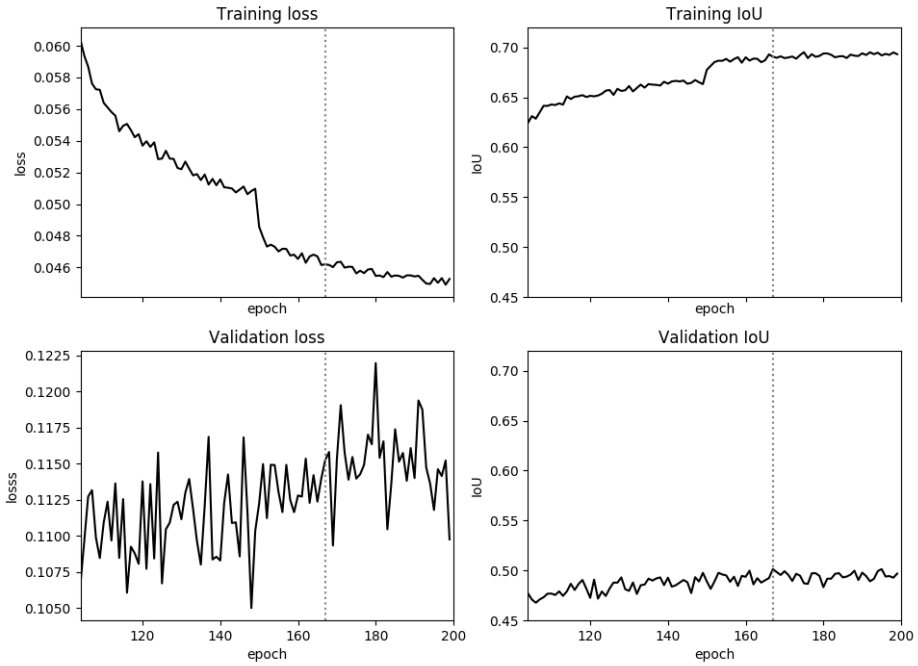

Figure A.7: Training plots for ENetLSTM-ksize1-summate.

Change loss

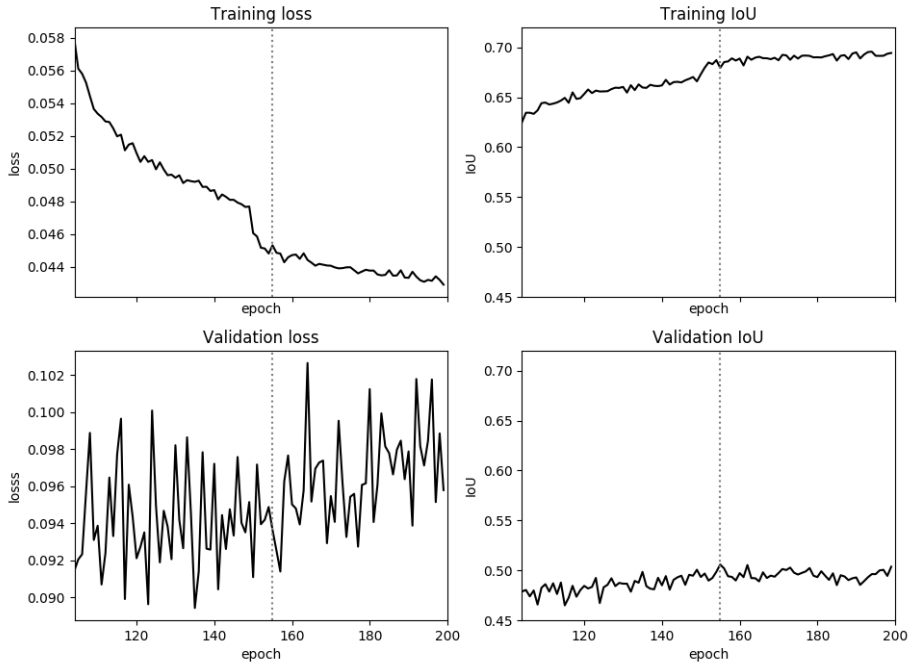

Figure A.8: Training plots for ENetLSTM-ksize1-summate with change loss.

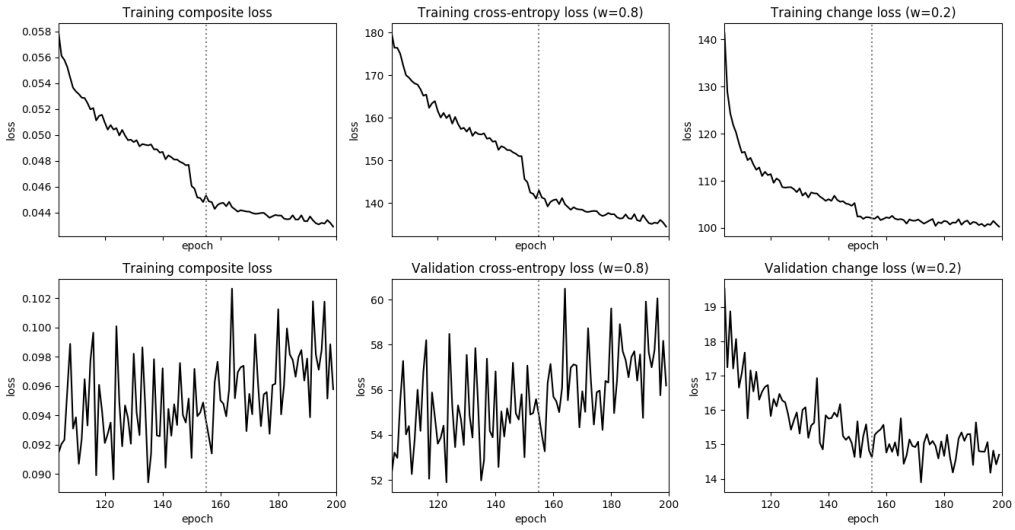

Figure A.9: The decomposition of the losses from figure A.8 into the cross-entropy and change loss. The cross-entropy loss has weight 0.8, and the change loss has weight 0.2.

Warping loss

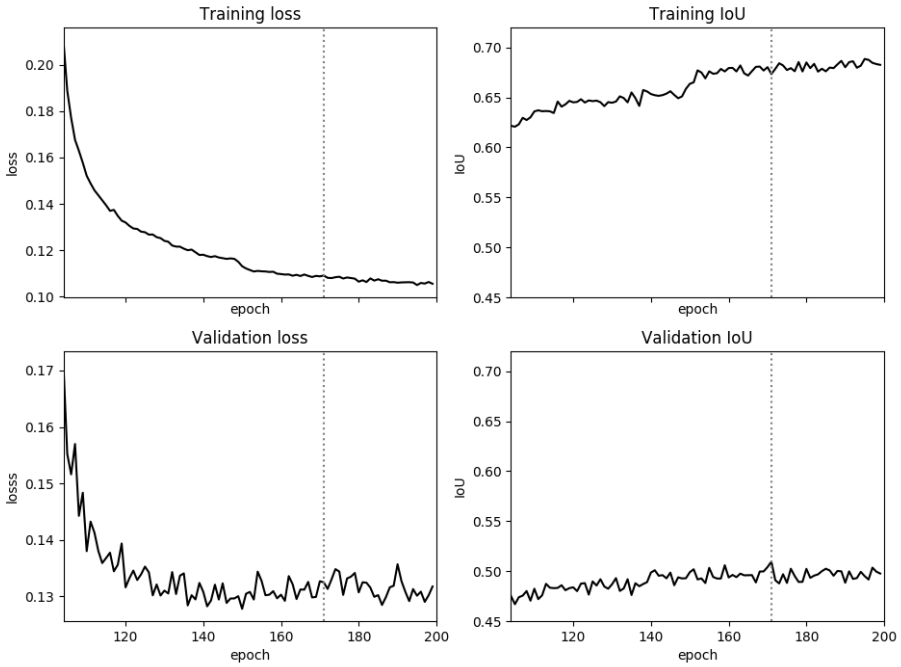

Figure A.10: Training plots for ENetLSTM-ksize1-summate with warping loss.

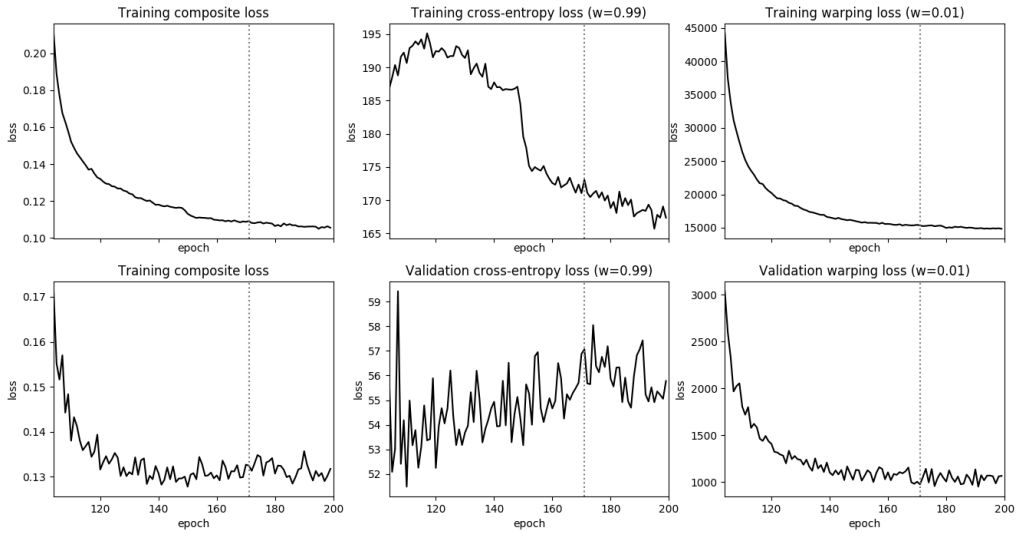

Figure A.11: The decomposition of the losses from figure A.10 into the cross-entropy and warping loss. The cross-entropy loss has weight 0.99, and the warping loss has weight 0.01.

A.2.3 ENetLSTM ksize3 concat

No temporal loss

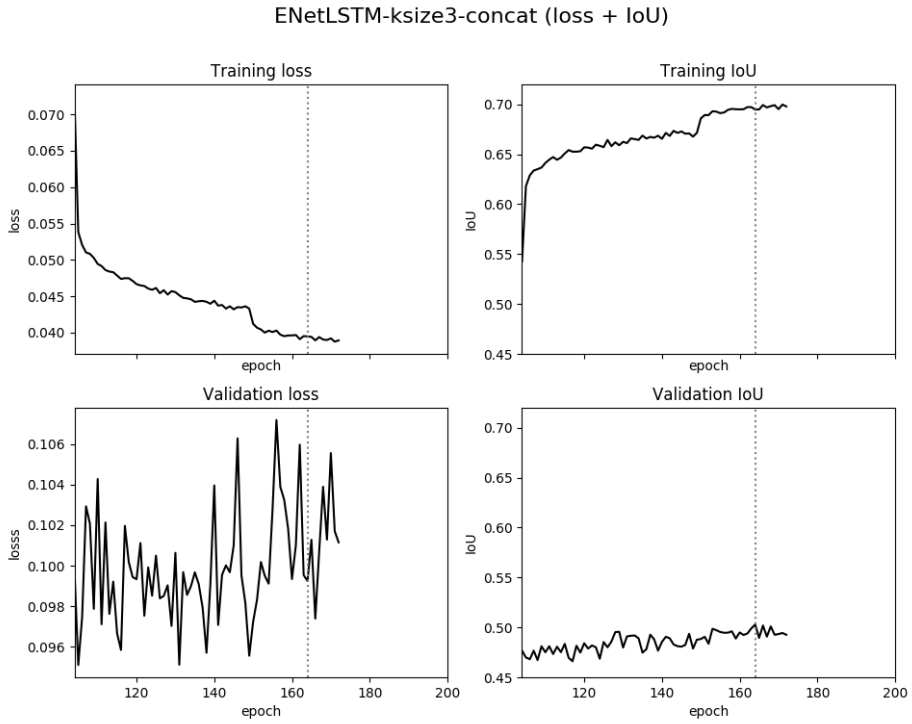

Figure A.12: Training plots for ENetLSTM-ksize3-concat. This network is only partially trained, up to epoch 172 instead of 200. Slightly better results may be possible when this network is trained further. However, it seems to be converged already.

Change loss

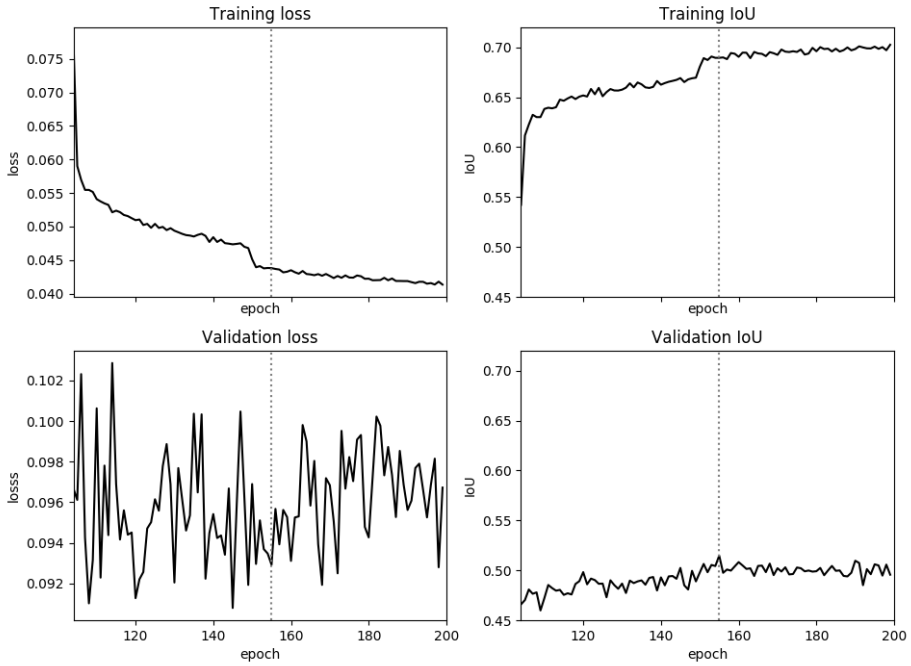

Figure A.13: Training plots for ENetLSTM-ksize3-concat with change loss.

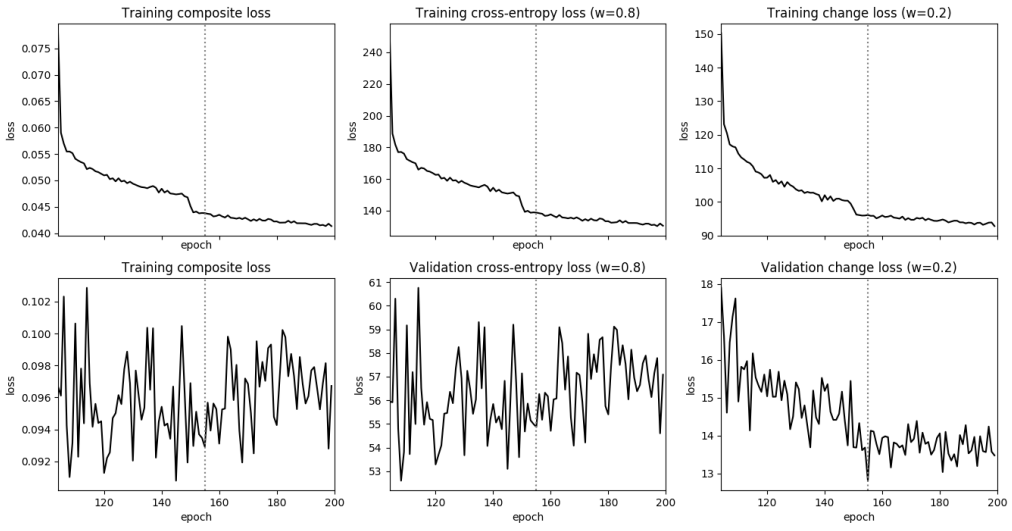

Figure A.14: The decomposition of the losses from figure A.13 into the cross-entropy and change loss. The cross-entropy loss has weight 0.8, and the change loss has weight 0.2.

Warping loss

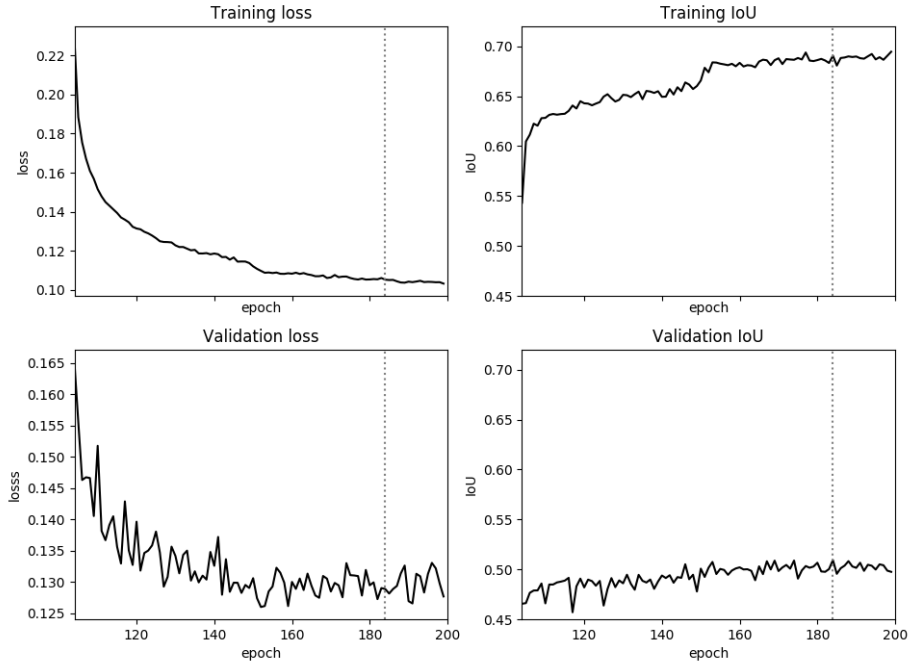

Figure A.15: Training plots for ENetLSTM-ksize3-concat with warping loss.

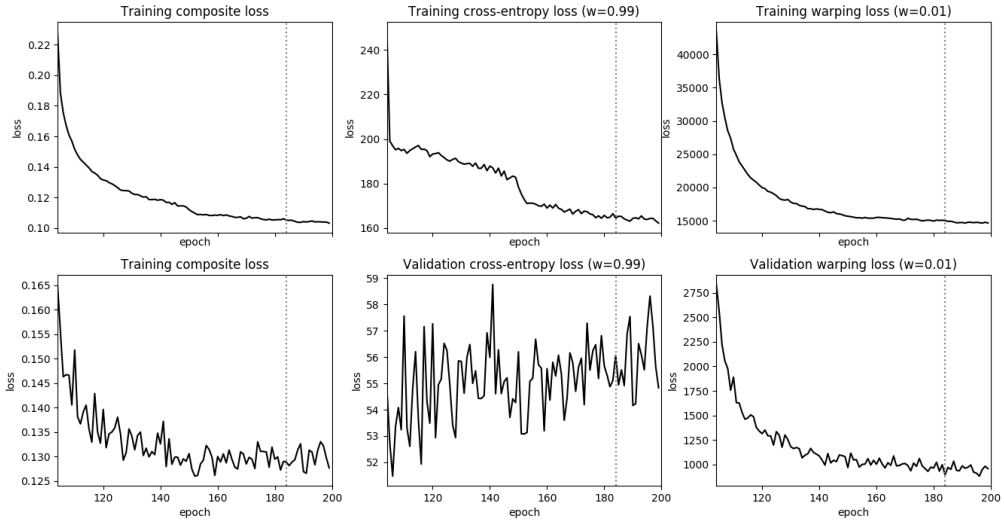

Figure A.16: The decomposition of the losses from figure A.15 into the cross-entropy and warping loss. The cross-entropy loss has weight 0.99, and the warping loss has weight 0.01.

A.2.4 ENetLSTM ksize3 summate

No temporal loss

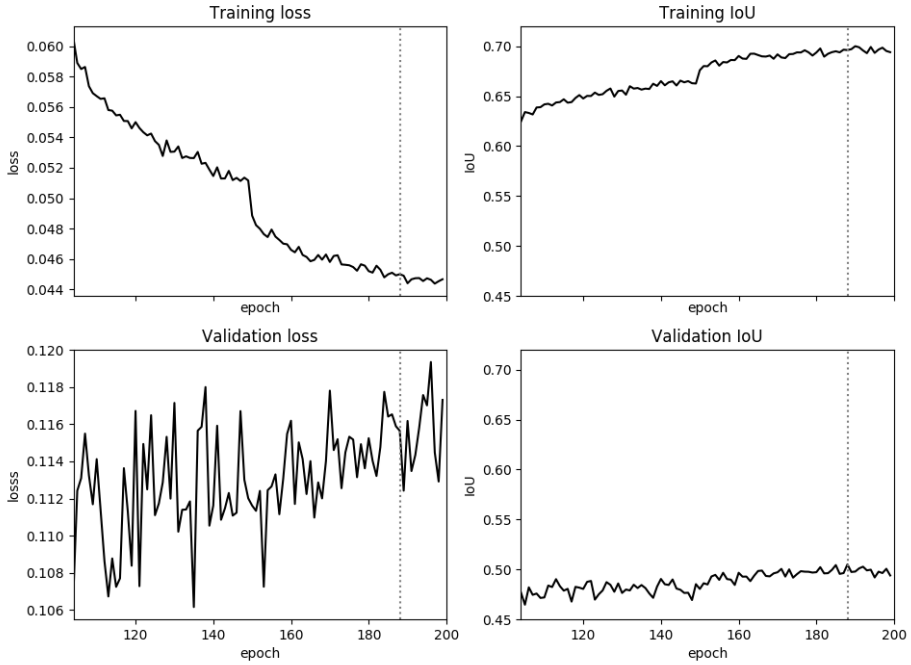

Figure A.17: Training plots for ENetLSTM-ksize3-summate.

Change loss

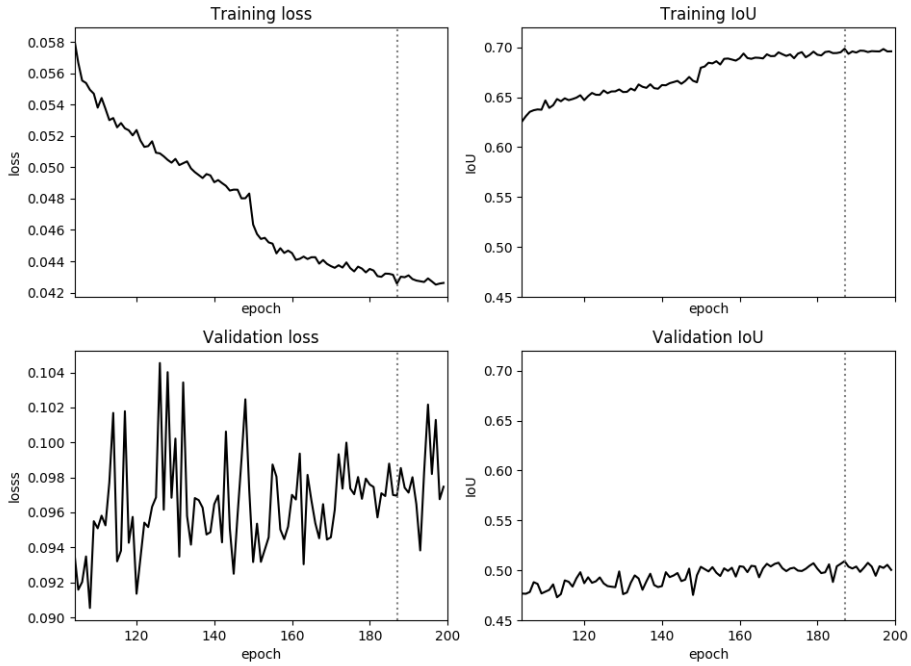

Figure A.18: Training plots for ENetLSTM-ksize3-summate with change loss.

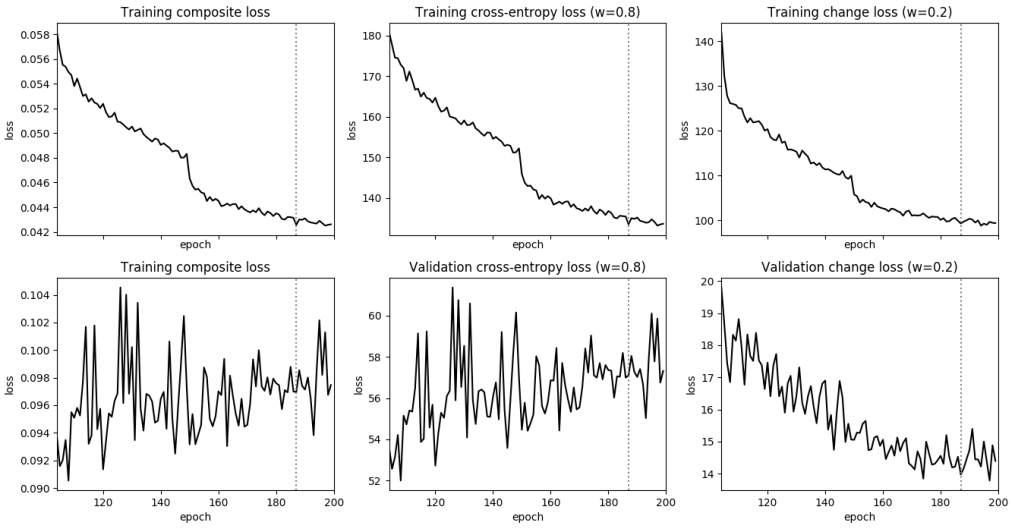

Figure A.19: The decomposition of the losses from figure A.18 into the cross-entropy and change loss. The cross-entropy loss has weight 0.8, and the change loss has weight 0.2.

Warping loss

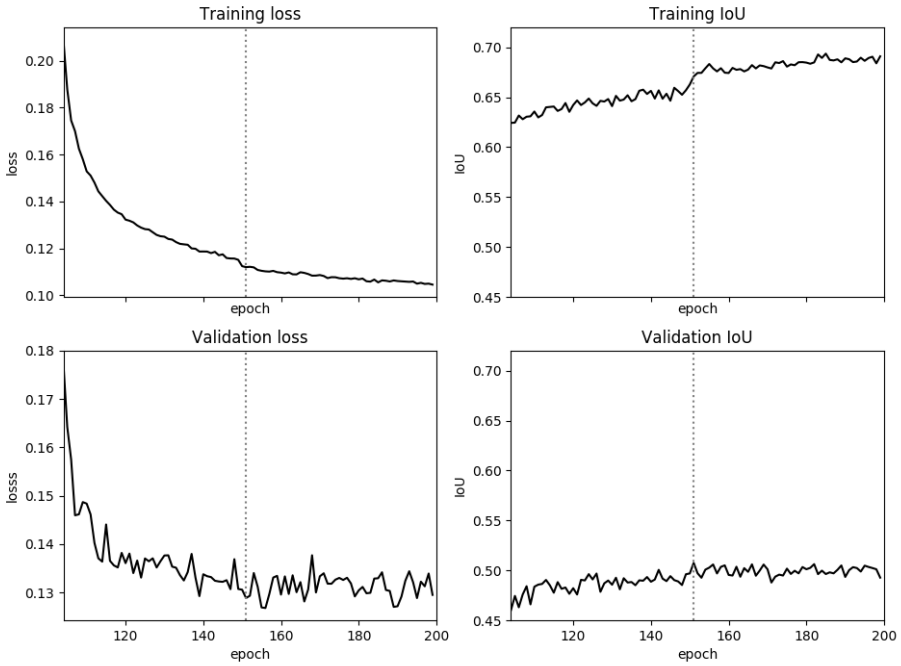

Figure A.20: Training plots for ENetLSTM-ksize3-summate with warping loss.

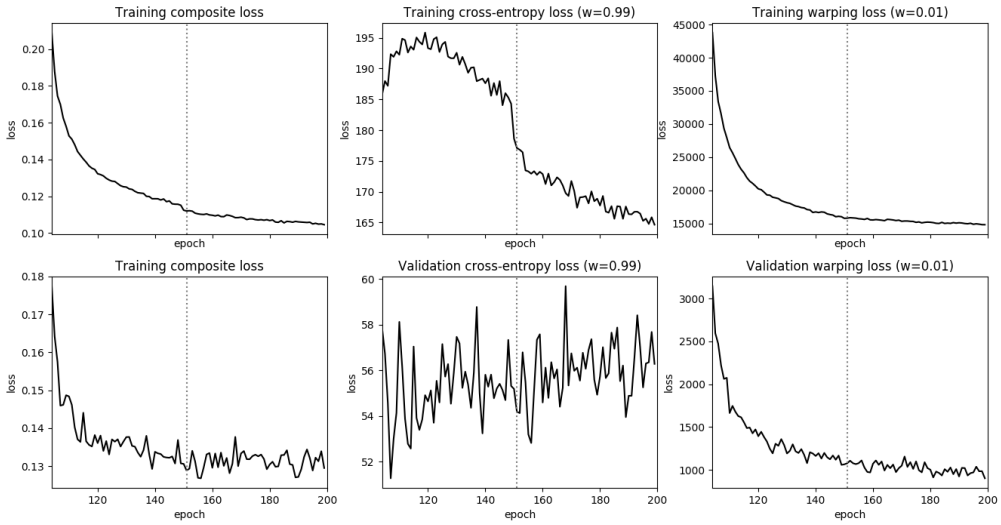

Figure A.21: The decomposition of the losses from figure A.20 into the cross-entropy and warping loss. The cross-entropy loss has weight 0.99, and the warping loss has weight 0.01.

A.3 ENetGRU architectures

A.3.1 ENetGRU ksize1 concat

No temporal loss

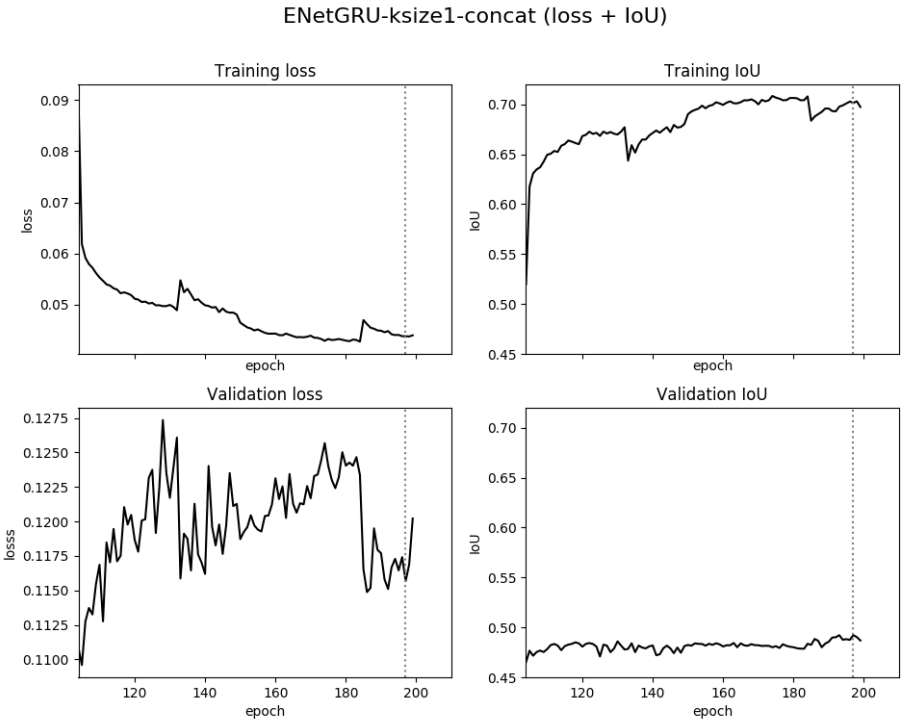

Figure A.22: Training plots for ENetGRU-ksize1-concat.

Change loss

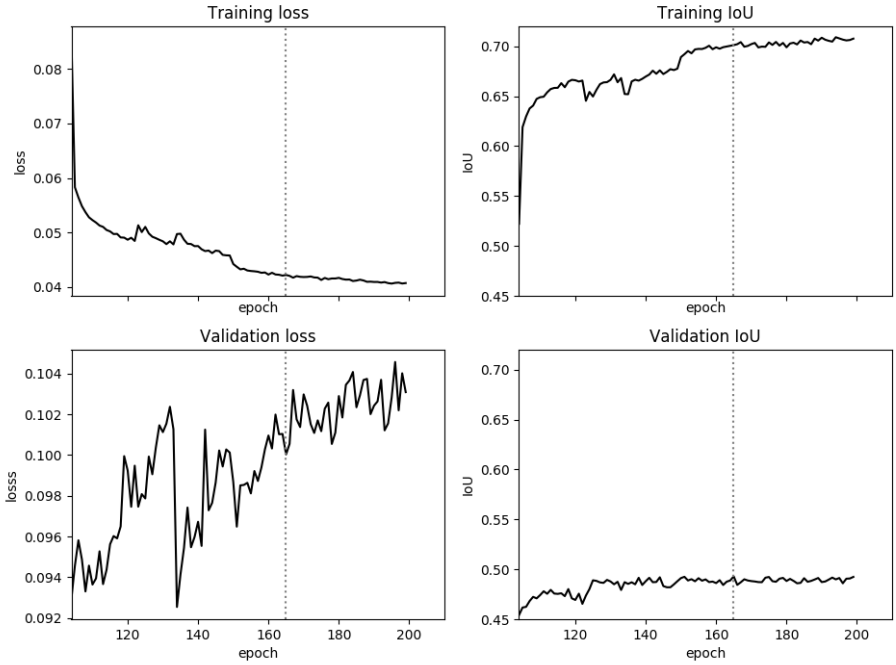

Figure A.23: Training plots for ENetGRU-ksize1-concat with change loss.

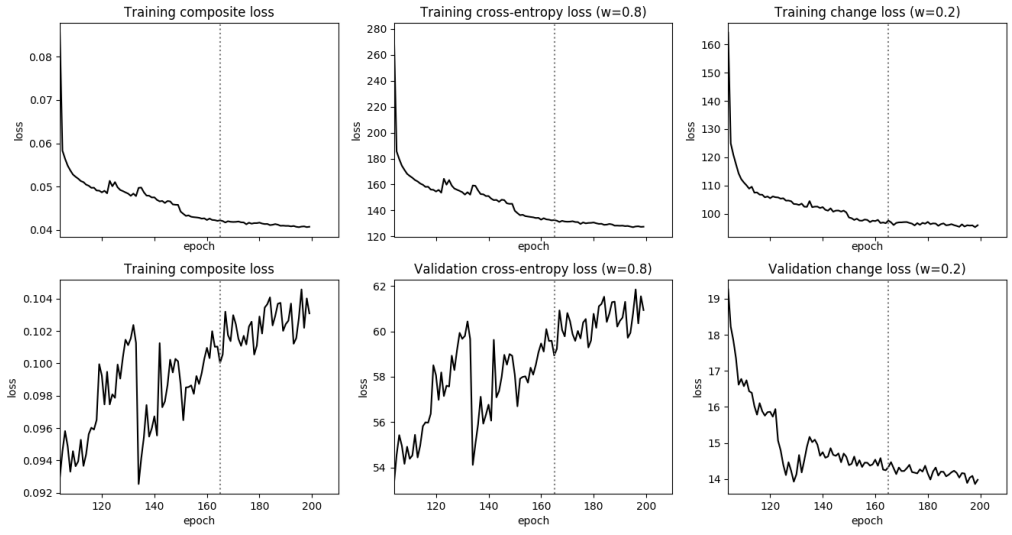

Figure A.24: The decomposition of the losses from figure A.23 into the cross-entropy and change loss. The cross-entropy loss has weight 0.8, and the change loss has weight 0.2.

Warping loss

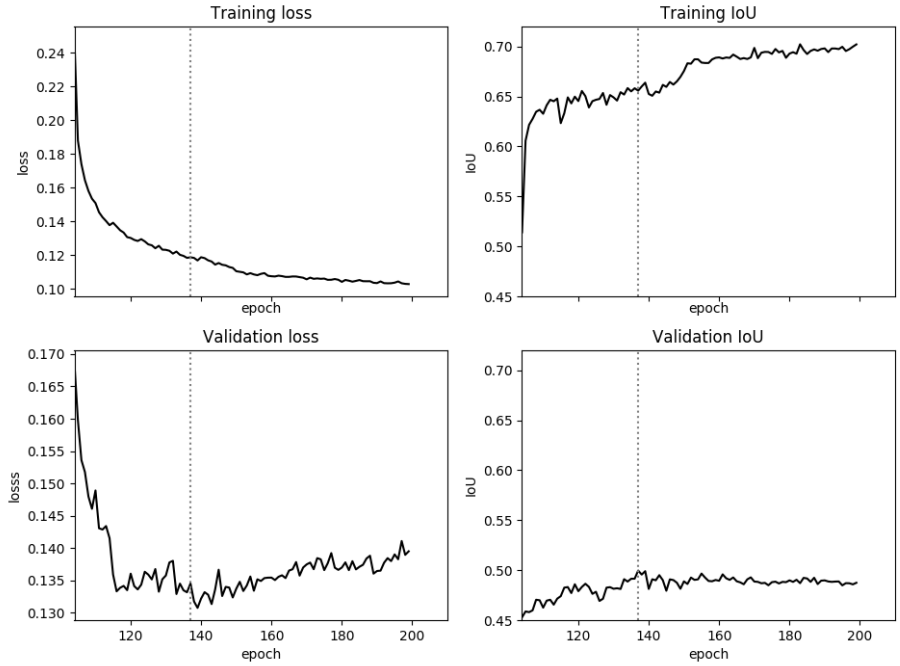

Figure A.25: Training plots for ENetGRU-ksize1-concat with warpingloss.

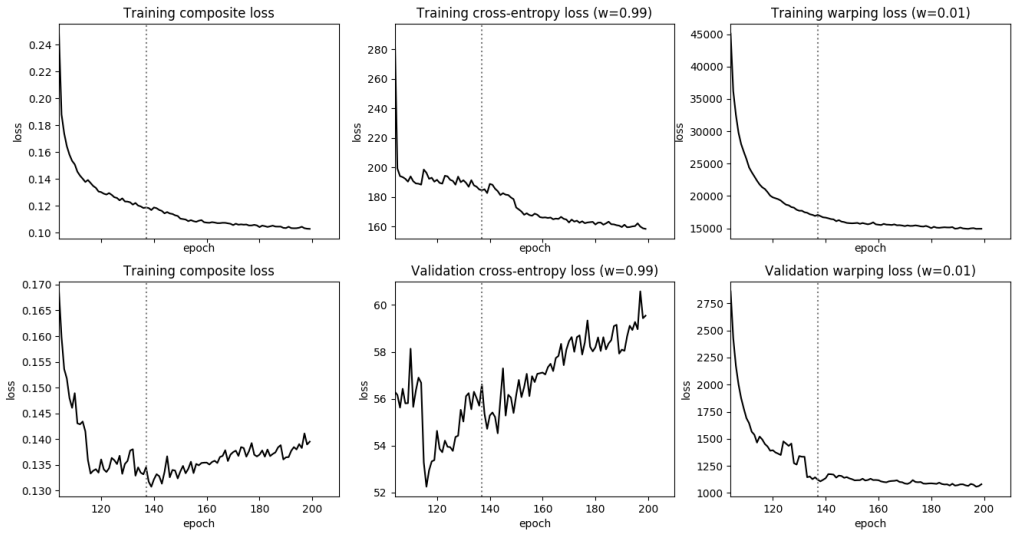

Figure A.26: The decomposition of the losses from figure A.25 into the cross-entropy and warping loss. The cross-entropy loss has weight 0.99, and the warping loss has weight 0.01.

A.3.2 ENetGRU ksize1 summate

No temporal loss

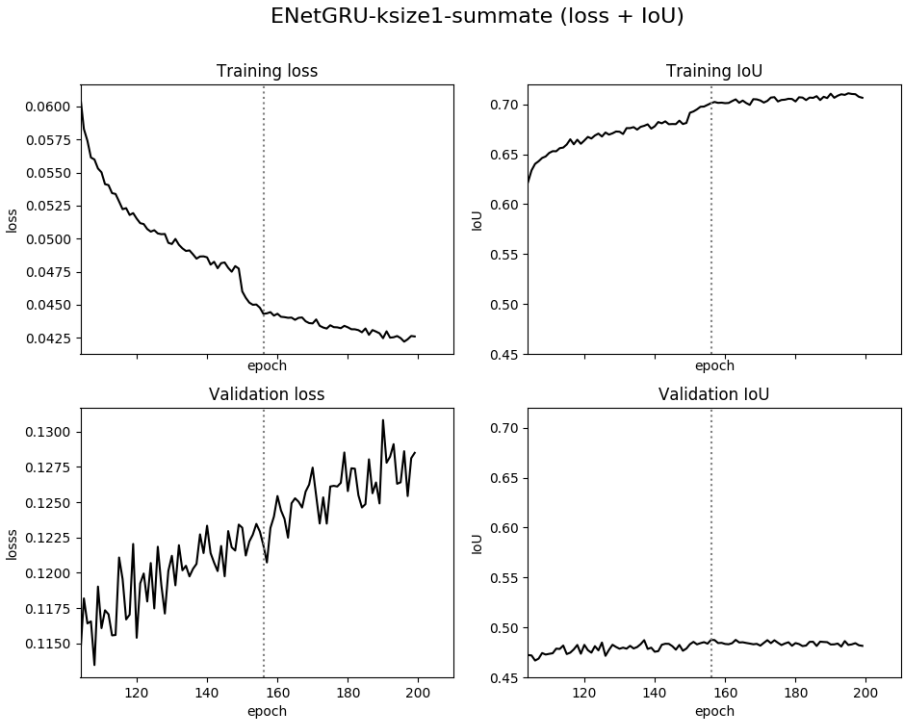

Figure A.27: Training plots for ENetGRU-ksize1-summate.

Change loss

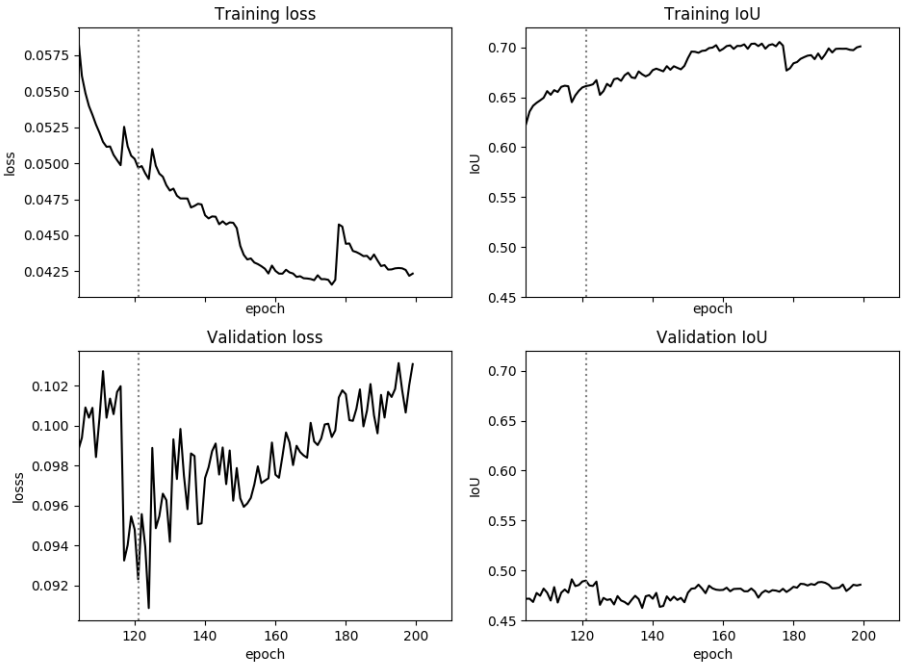

Figure A.28: Training plots for ENetGRU-ksize1-summate with change loss.

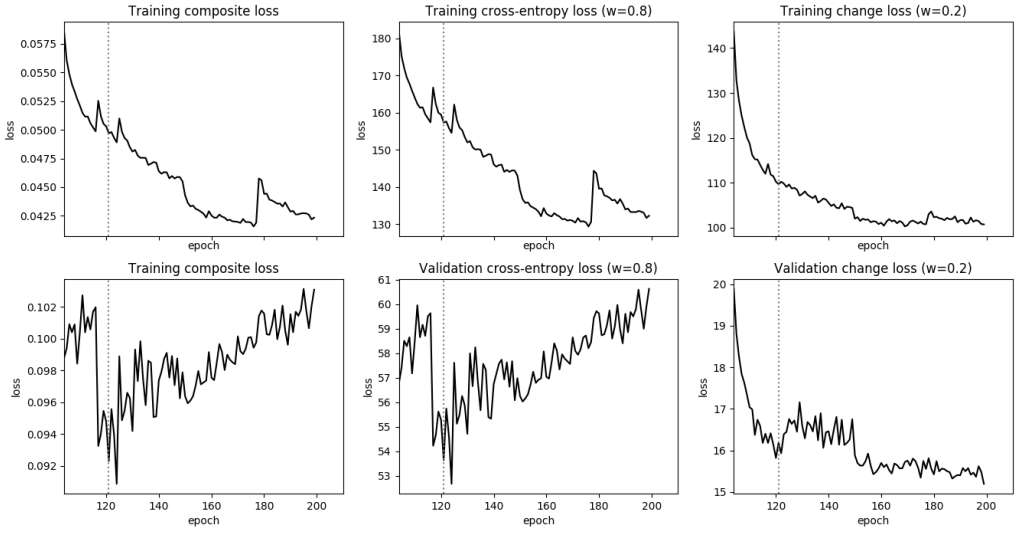

Figure A.29: The decomposition of the losses from figure A.28 into the cross-entropy and change loss. The cross-entropy loss has weight 0.8, and the change loss has weight 0.2.

Warping loss

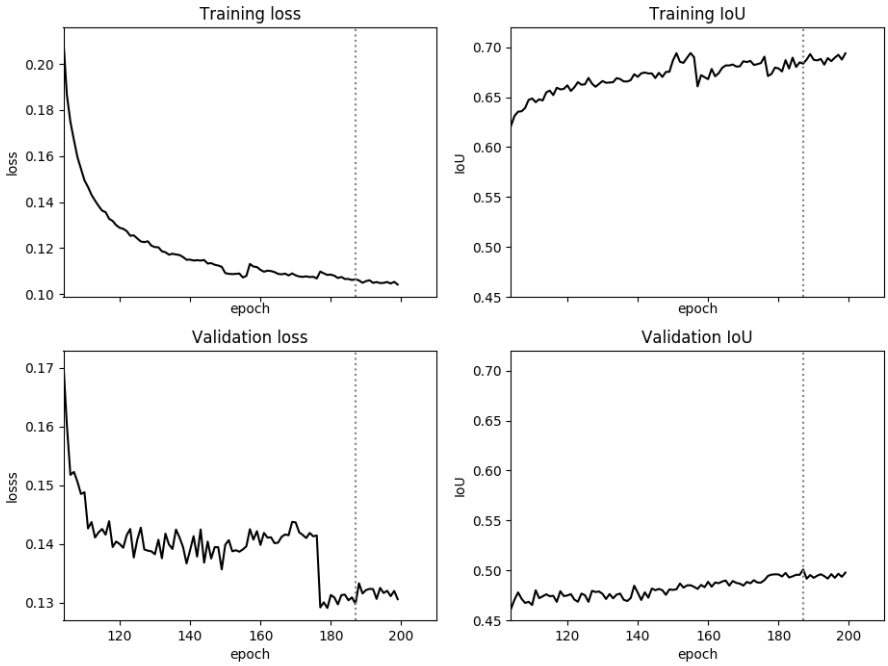

Figure A.30: Training plots for ENetGRU-ksize1-summate with warping loss.

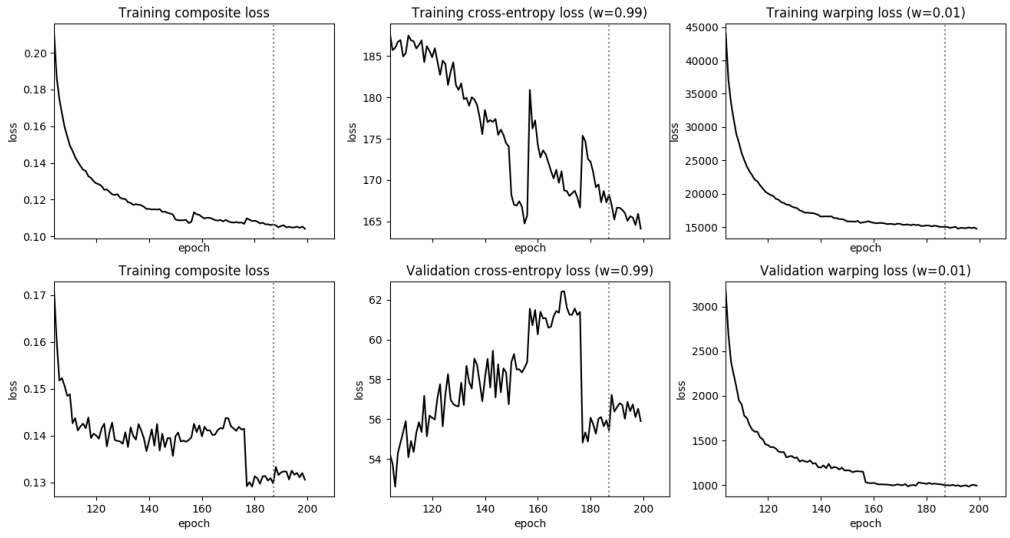

Figure A.31: The decomposition of the losses from figure A.30 into the cross-entropy and warping loss. The cross-entropy loss has weight 0.99, and the warping loss has weight 0.01.

A.3.3 ENetGRU ksize3 concat

No temporal loss

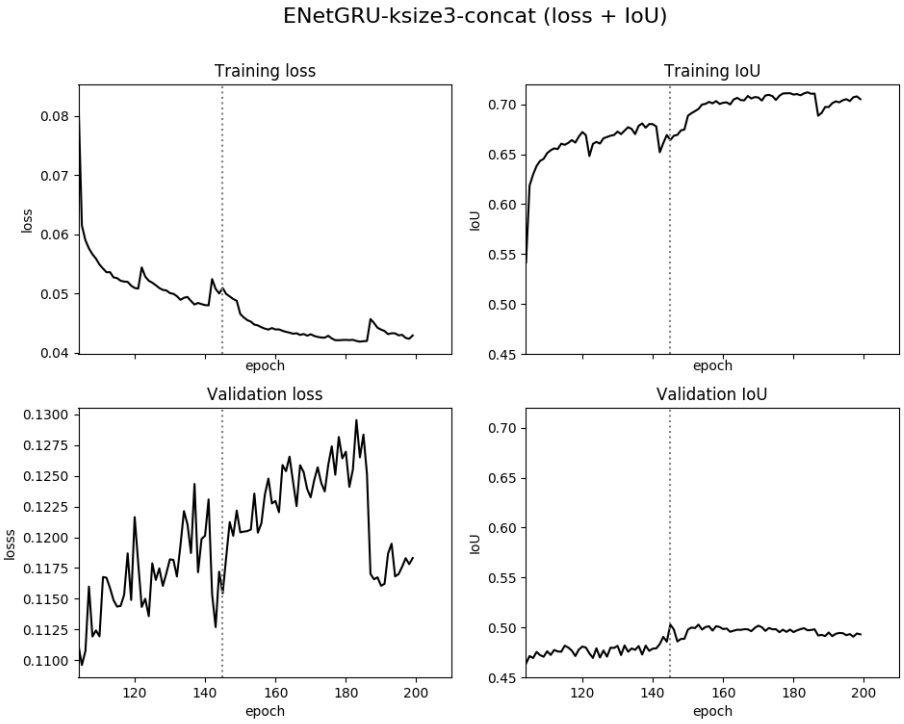

Figure A.32: Training plots for ENetGRU-ksize3-concat.

Change loss

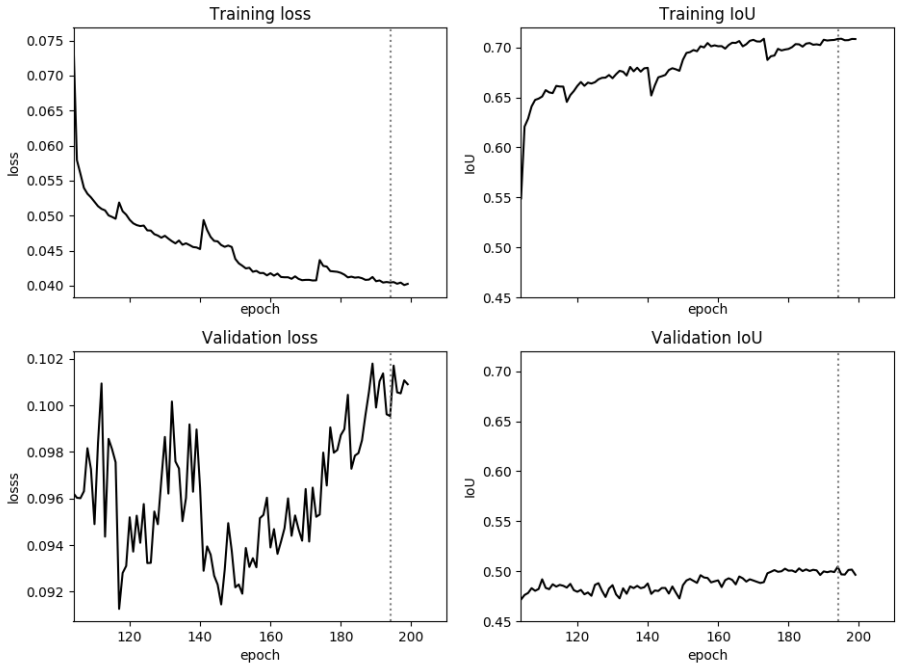

Figure A.33: Training plots for ENetGRU-ksize3-concat with change loss.

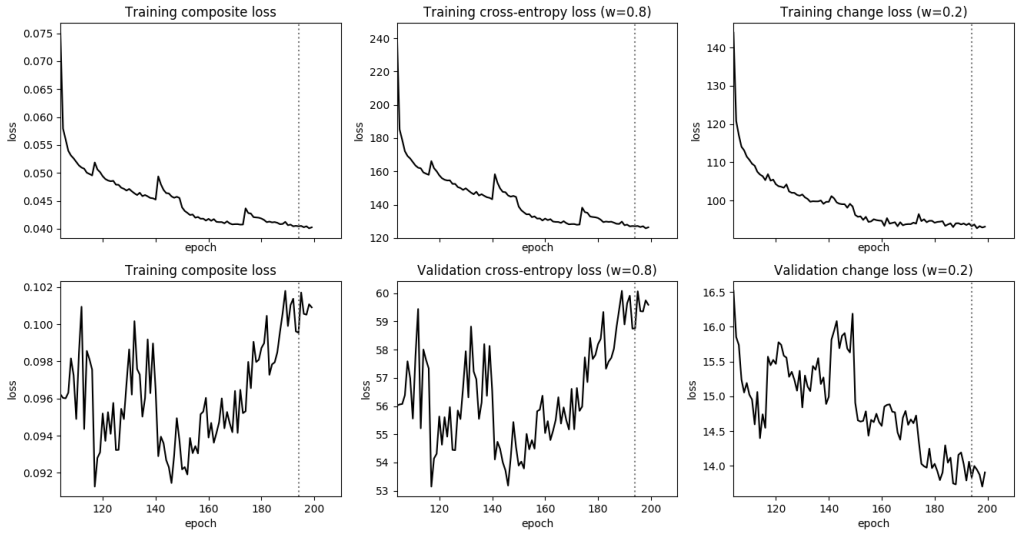

Figure A.34: The decomposition of the losses from figure A.33 into the cross-entropy and change loss. The cross-entropy loss has weight 0.8, and the change loss has weight 0.2.

Warping loss

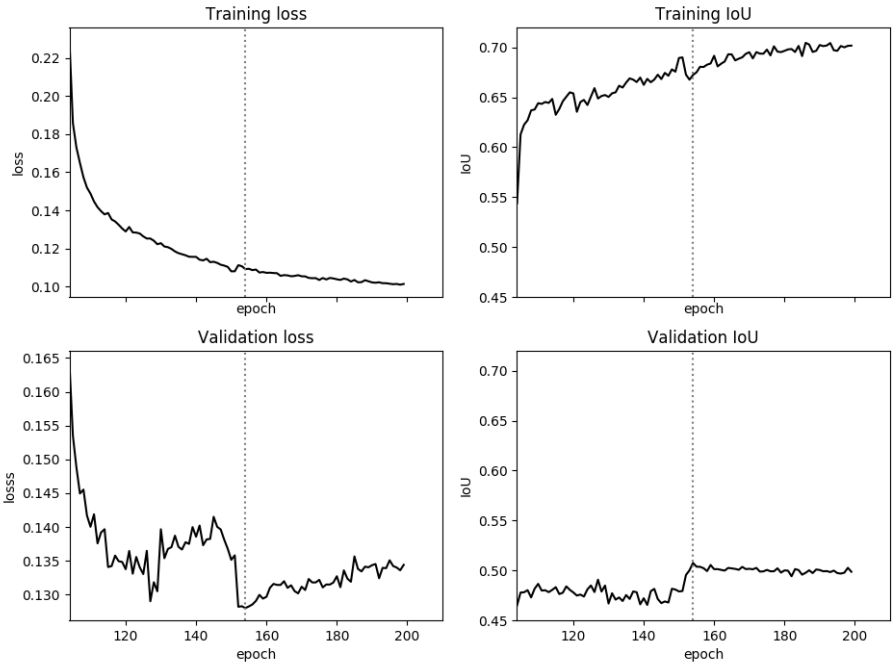

Figure A.35: Training plots for ENetGRU-ksize3-concat with warping loss.

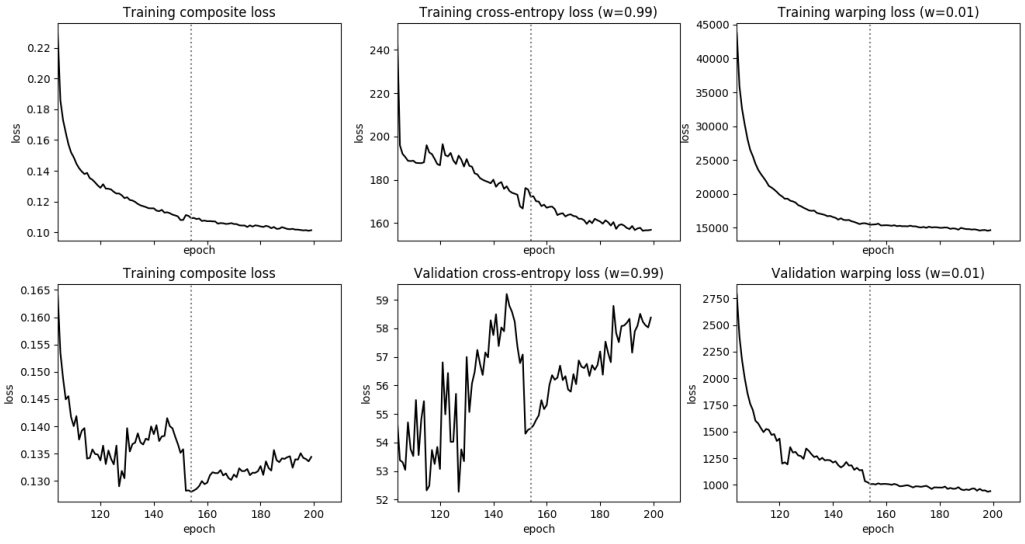

Figure A.36: The decomposition of the losses from figure A.35 into the cross-entropy and warping loss. The cross-entropy loss has weight 0.99, and the warping loss has weight 0.01.

A.3.4 ENetGRU ksize3 summate

No temporal loss

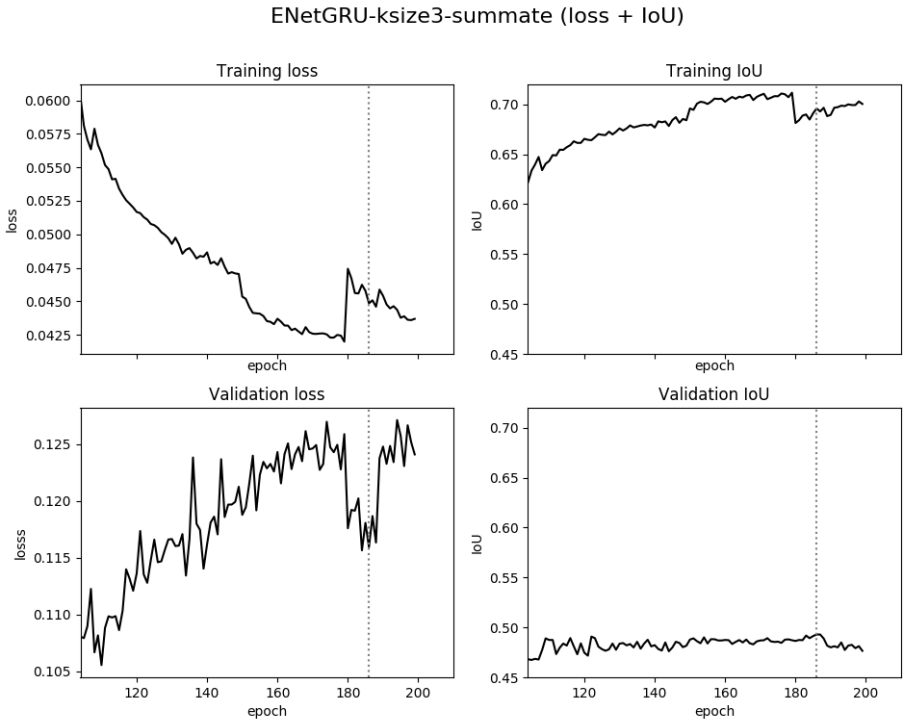

Figure A.37: Training plots for ENetGRU-ksize3-summate.

Change loss

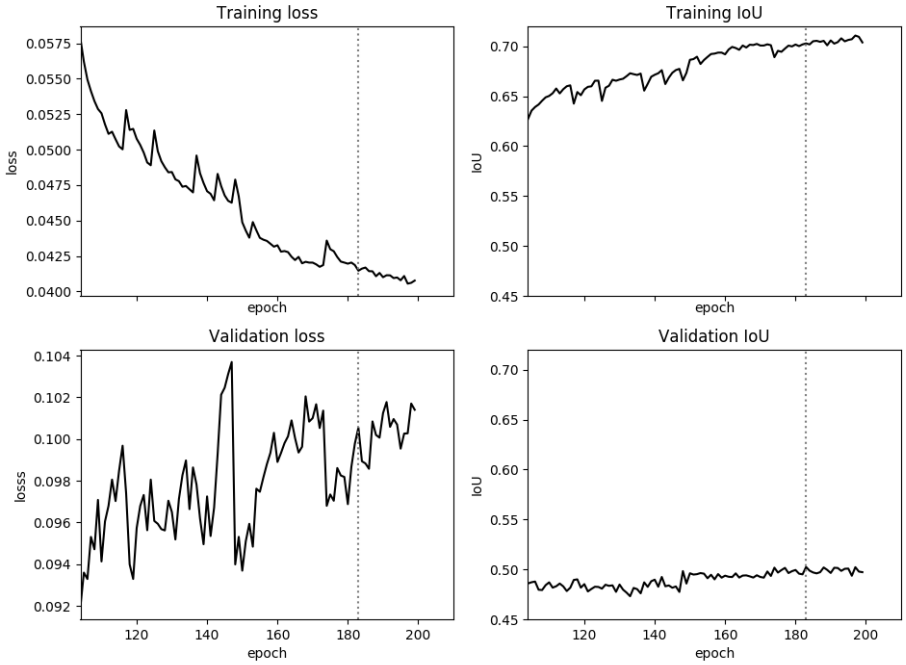

Figure A.38: Training plots for ENetGRU-ksize3-summate with change loss.

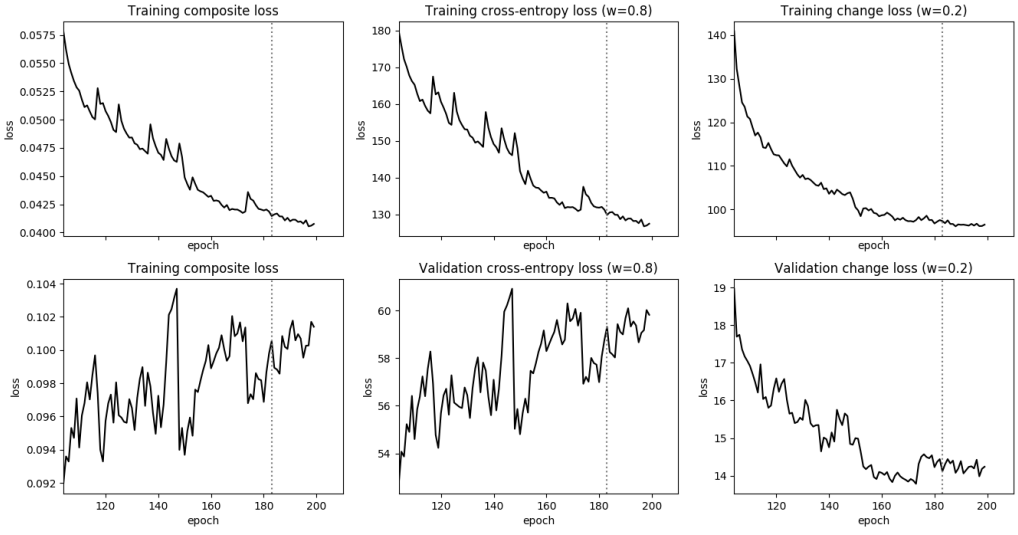

Figure A.39: The decomposition of the losses from figure A.38 into the cross-entropy and change loss. The cross-entropy loss has weight 0.8, and the change loss has weight 0.2.

Warping loss

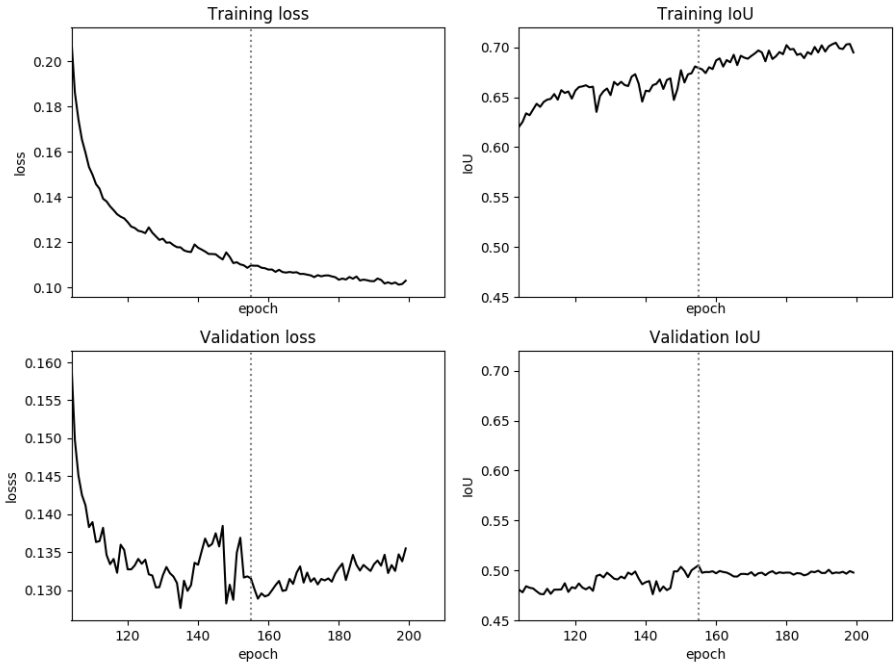

Figure A.40: Training plots for ENetGRU-ksize3-summate with warping loss.

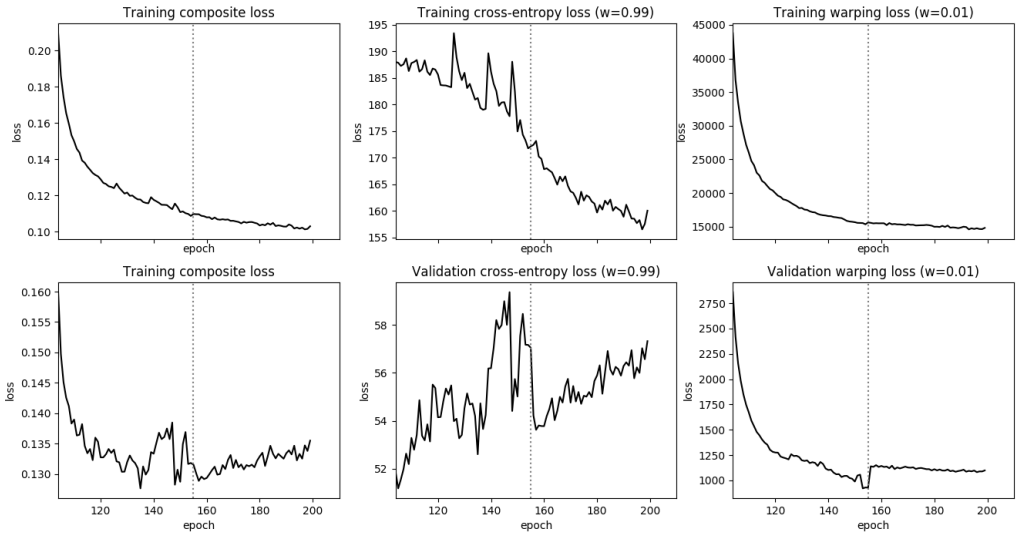

Figure A.41: The decomposition of the losses from figure A.40 into the cross-entropy and warping loss. The cross-entropy loss has weight 0.99, and the warping loss has weight 0.01.

Appendix B

Prior Knowledge and Own Work

In this chapter, I shortly describe my prior knowledge of this topic and what parts of this thesis I implemented myself. Hopefully, this provides an accurate view on the amount of time and energy I invested during the year.

Before this thesis, I had followed a few elective courses about computer vision, such as *Pattern Recognition and Image Interpretation* and *Computer Vision*. However, these courses mainly focused on the classical computer vision methods without deep learning. So at the start of this thesis, I had not implemented a neural network yet and only had a superficial idea about its working. Therefore, the first months consisted of reading through papers and blogs to learn the working of neural networks and to learn the PyTorch framework. During this literature study, I kept a structured document with small summaries of the papers I read, to be able to retain an overview of the large amount of research that has been done on this topic. This allowed to write a fairly extensive *Related Work* section.

Because PyTorch has not a lot of support for video processing yet, I wrote a large part of the framework myself. I started from the ENet implementation on the Github repository of David Silva [73], and gradually extended it to support video files, recurrent neural networks and training sessions with multiple parameter variations. The support of videos or image sequences required writing custom data loaders. After all, large image sequences cannot be loaded into memory at once and therefore must be loaded per set of frames. To be able to train on mini-batches of video data with batch size N, these loaders load the frames of N different videos in parallel, and every i-th frame of the videos is placed in one batch. The full framework supports videos of different lengths being taken together in batches, and supports video file loading in separate threads. When RNNs then come into play, the networks, loss functions and metrics need an internal state to bridge the gaps between the batches of frames. Indeed, since the frames of a video are loaded into subsequent batches, the temporal information must be kept when the batches advance through the network. However, when a batch contains the first frame of a new independent sequence, all states must be reset. Another difficulty related to these states, is the training with

backpropagation through time. Backpropagation through time implies that the loss must be backpropagated back through one or more batch "boundaries", as one batch represent one timestep. This means that not only the state of the network, but also the calculation history of this state must be managed carefully. This support for videos and RNNs is entirely my own work. It was definitely the hardest part of the thesis. And although improvements are certainly still possible, the framework architecture has improved a lot over time. The source code is available online on Github: https://github.com/JohannesVerherstraeten/semantic-video-segmentation.

Next to the data loading part of the framework, I also included an implementation of the ConvLSTM and ConvGRU, based on the repositories of Andrea Palazzi [61] and Jacob Kimmel [39] respectively. The warping loss implementation is based on the paper of [43] and the standard IoU is copied from the same repository from David Silva [73] as ENet. The code structure allowing the description of a training session using a configuration file, is based on the PyTorch template from Victor Huang [35]. Finally, the change loss, smoothing loss and temporal IoU metric are proposed by this thesis, so were fully implemented by me.

Bibliography

[1] V. Badrinarayanan, A. Kendall, and R. Cipolla. SegNet: A Deep Convolutional Encoder-Decoder Architecture for Image Segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2017.

[2] N. Ballas, L. Yao, C. Pal, and A. Courville. Delving Deeper into Convolutional Networks for Learning Video Representations. *arXiv:1511.06432 [cs.CV]*, 2015.

[3] Y. Bengio, P. Simard, and P. Frasconi. Learning Long-Term Dependencies with Gradient Descent is Difficult. *IEEE Transactions on Neural Networks*, 1994.

[4] M. D. Binder, N. Hirokawa, and U. Windhorst. Aperture Problem. In *Encyclopedia of Neuroscience*. Springer Berlin Heidelberg, 2008.

[5] N. Bonneel, J. Tompkin, K. Sunkavalli, D. Sun, S. Paris, and H. Pfister. Blind Video Temporal Consistency. *ACM Transactions on Graphics*, 2015.

[6] G. J. Brostow, J. Fauqueur, and R. Cipolla. Semantic Object Classes in Video: A High-Definition Ground Truth Database. *Pattern Recognition Letters*, 2009.

[7] G. J. Brostow, J. Shotton, J. Fauqueur, and R. Cipolla. Segmentation and Recognition Using Structure from Motion Point Clouds. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 2008.

[8] W. Byeon, T. M. Breuel, F. Raue, and M. Liwicki. Scene Labeling with LSTM Recurrent Neural Networks. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015.

[9] S. Caelles, A. Montes, K.-K. Maninis, Y. Chen, L. Van Gool, F. Perazzi, and J. Pont-Tuset. The 2018 DAVIS Challenge on Video Object Segmentation. *arXiv:1803.00557 [cs.CV]*, 2018.

[10] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille. Semantic Image Segmentation with Deep Convolutional Nets and Fully Connected CRFs. In *International Conference on Learning Representation (ICLR)*, 2015.

[11] L.-C. 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. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2018.

[12] L.-C. Chen, G. Papandreou, F. Schroff, and H. Adam. Rethinking Atrous Convolution for Semantic Image Segmentation. *arXiv:1706.05587 [cs.CV]*, 2017.

[13] K. Cho, B. van Merrienboer, D. Bahdanau, and Y. Bengio. On the Properties of Neural Machine Translation: Encoder-Decoder Approaches. *arXiv:1409.1259 [cs.CL]*, 2014.

[14] K. Cho, B. van Merrienboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio. Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation. *arXiv:1406.1078 [cs.CL]*, 2014.

[15] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio. Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling. *arXiv:1412.3555 [cs.NE]*, 2014.

[16] D. C. Ciresan, A. Giusti, L. M. Gambardella, and J. Schmidhuber. Mitosis Detection in Breast Cancer Histology Images with Deep Neural Networks. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 2013.

[17] D. C. Ciresan, U. Meier, and J. Schmidhuber. Multi-Column Deep Neural Networks for Image Classification. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2012.

[18] M. Cordts, M. Omran, S. Ramos, T. Rehfeld, M. Enzweiler, R. Benenson, U. Franke, S. Roth, and B. Schiele. The Cityscapes Dataset for Semantic Urban Scene Understanding. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2016.

[19] J. Deng, W. Dong, R. Socher, L.-J. Li, Kai Li, and Li Fei-Fei. ImageNet: A Large-Scale Hierarchical Image Database. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2009.

[20] M. Everingham, L. Van Gool, C. K. Williams, J. Winn, and A. Zisserman. The PASCAL Visual Object Classes (VOC) Challenge. *International Journal of Computer Vision*, 2010.

[21] M. Fayyaz, M. H. Saffar, M. Sabokrou, M. Fathy, F. Huang, and R. Klette. STFCN: Spatio-Temporal Fully Convolutional Neural Network for Semantic Segmentation of Street Scenes. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 2017.

[22] K. Fukushima. Neocognitron: A Self-Organizing Neural Network Model for a Mechanism of Pattern Recognition Unaffected By Shift in Position. *Biological Cybernetics*, 1980.

[23] R. Gadde, V. Jampani, and P. V. Gehler. Semantic Video CNNs Through Representation Warping. In *Proceedings of the IEEE International Conference on Computer Vision*, 2017.

[24] F. Gers and J. Schmidhuber. Recurrent Nets that Time and Count. In *Proceedings of the IEEE-INNS-ENNS International Joint Conference on Neural Networks. IJCNN 2000. Neural Computing: New Challenges and Perspectives for the New Millennium*, 2000.

[25] X. Glorot and Y. Bengio. Understanding the Difficulty of Training Deep Feedforward Neural Networks. In *Proceedings of the International Conference on Artificial Intelligence and Statistics*, 2010.

[26] X. Glorot, A. Bordes, and Y. Bengio. Deep Sparse Rectifier Neural Networks. *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, 2011.

[27] R. Gomez. Understanding Categorical Cross-Entropy Loss, Binary Cross-Entropy Loss, Softmax Loss, Logistic Loss, Focal Loss and all those confusing names. https://gombru.github.io/2018/05/23/cross_entropy_loss/. Accessed on 2019-04-05, 2018.

[28] M. A. Goodale and A. D. Milner. Separate Visual Pathways for Perception and Action. *Trends in Neurosciences*, 1992.

[29] A. Graves, M. Liwicki, S. Fernández, R. Bertolami, H. Bunke, and J. Schmidhuber. A Novel Connectionist System for Unconstrained Handwriting Recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2009.

[30] K. Greff, R. K. Srivastava, J. Koutnik, B. R. Steunebrink, and J. Schmidhuber. LSTM: A Search Space Odyssey. *IEEE Transactions on Neural Networks and Learning Systems*, 2017.

[31] P. Grother. NIST Special Database: 19 Handprinted Forms and Characters Database. Technical report, 1995.

[32] K. He, X. Zhang, S. Ren, and J. Sun. Deep Residual Learning for Image Recognition. *arXiv:1512.03385 [cs.CV]*, 2015.

[33] S. Hochreiter and J. Schmidhuber. Long Short-Term Memory. *Neural Computation*, 1997.

[34] M. Holschneider, R. Kronland-Martinet, J. Morlet, and P. Tchamitchian. A Real-Time Algorithm for Signal Analysis with the Help of the Wavelet Transform. In J.-M. Combes, A. Grossmann, and P. Tchamitchian, editors, *Wavelets. Inverse Problems and Theoretical Imaging.* Springer Berlin Heidelberg, 1990.

[35] V. Huang. Github page of PyTorch Template. https://github.com/victoresque/pytorch-template. Accessed on 2018-11-16., 2018.

[36] E. Ilg, N. Mayer, T. Saikia, M. Keuper, A. Dosovitskiy, and T. Brox. FlowNet 2.0: Evolution of Optical Flow Estimation with Deep Networks. *arXiv:1612.01925 [cs.CV]*, 2016.

[37] S. Ioffe and C. Szegedy. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. *arXiv:1502.03167 [cs.LG]*, 2015.

[38] S. D. Jain, B. Xiong, and K. Grauman. FusionSeg: Learning to Combine Motion and Appearance for Fully Automatic Segmentation of Generic Objects in Videos. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[39] J. Kimmel. Github Page of Pytorch ConvGRU. https://github.com/jacobkimmel/pytorch_convgru. Accessed on 2019-05-14., 2017.

[40] D. P. Kingma and J. Ba. Adam: A Method for Stochastic Optimization. *arXiv:1412.6980 [cs.LG]*, dec 2014.

[41] P. Krähenbühl and V. Koltun. Efficient Inference in Fully Connected CRFs with Gaussian Edge Potentials[Slices]. *Advances in Neural Information Processing Systems (NIPS)*, 2011.

[42] A. Krizhevsky, I. Sutskever, and G. E. Hinton. ImageNet Classification with Deep Convolutional Neural Networks. In *Advances in Neural Information Processing Systems (NIPS)*, 2012.

[43] W. S. Lai, J. B. Huang, O. Wang, E. Shechtman, E. Yumer, and M. H. Yang. Learning Blind Video Temporal Consistency. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 2018.

[44] Y. LeCun, Y. Bengio, and Others. Convolutional Networks for Images, Speech, and Time Series. *The Handbook of Brain Theory and Neural Networks*, 1995.

[45] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-Based Learning Applied to Document Recognition. *Proceedings of the IEEE*, 1998.

[46] Y. LeCun, Fu Jie Huang, and L. Bottou. Learning Methods for Generic Object Recognition with Invariance to Pose and Lighting. In *Computer Vision and Pattern Recognition*, 2004.

[47] A. Lenail. NN-SVG: Publication-Ready NN-Architecture Schematics. http://alexlenail.me/NN-SVG/AlexNet.html. Accessed on 2019-05-15.

[48] X. Li and X. Wu. Constructing Long Short-Term Memory Based Deep Recurrent Neural Networks for Large Vocabulary Speech Recognition. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2015.

[49] Z. Li, E. Gavves, M. Jain, and C. G. M. Snoek. VideoLSTM Convolves, Attends and Flows for Action Recognition. *Computer Vision and Image Understanding*, 2016.

[50] G. Lin, A. Milan, C. Shen, and I. Reid. RefineNet: Multi-path refinement networks for high-resolution semantic segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[51] J. Long, E. Shelhamer, and T. Darrell. Fully Convolutional Networks for Semantic Segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2015.

[52] B. D. Lucas and T. Kanade. An Iterative Image Registration Technique with an Application to Stereo Vision. In *Proceedings of the 7th International Joint Conference on Artificial Intelligence (IJCAI)*, 1981.

[53] M.-T. Luong, H. Pham, and C. D. Manning. Effective Approaches to Attention-based Neural Machine Translation. *arXiv:1508.04025 [cs.CL]*, 2015.

[54] S. Mehta, M. Rastegari, A. Caspi, L. Shapiro, and H. Hajishirzi. ESPNet: Efficient Spatial Pyramid of Dilated Convolutions for Semantic Segmentation. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 2018.

[55] G. Montufar, R. Pascanu, K. Cho, and Y. Bengio. On the Number of Linear Regions of Deep Neural Networks. *Advances in Neural Information Processing Systems (NIPS)*, 2014.

[56] J. Y. H. Ng, M. Hausknecht, S. Vijayanarasimhan, O. Vinyals, R. Monga, and G. Toderici. Beyond Short Snippets: Deep Networks for Video Classification. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015.

[57] D. Nilsson and C. Sminchisescu. Semantic Video Segmentation by Gated Recurrent Flow Propagation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[58] H. Noh, S. Hong, and B. Han. Learning Deconvolution Network for Semantic Segmentation. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015.

[59] C. Olah. Understanding LSTM Networks - Colah's blog. http://colah.github.io/posts/2015-08-Understanding-LSTMs/. Accessed on 2019-05-09.

[60] Oxford University Press. Definition of Overfitting by the Oxford English Dictionary. https://en.oxforddictionaries.com/definition/overfitting. Accessed on 2019-05-03.

[61] A. Palazzi. Github Page of PyTorch ConvLSTM. https://github.com/ndrplz/ConvLSTM_pytorch. Accessed on 2019-05-14., 2018.

[62] R. Pascanu, T. Mikolov, and Y. Bengio. On the Difficulty of Training Recurrent Neural Networks. In *International Conference on Machine Learning*, 2013.

[63] A. Paszke, A. Chaurasia, S. Kim, and E. Culurciello. ENet: A Deep Neural Network Architecture for Real-Time Semantic Segmentation. *arXiv:1606.02147 [cs.CV]*, 2016.

[64] C. Peng, X. Zhang, G. Yu, G. Luo, and J. Sun. Large Kernel Matters - Improve Semantic Segmentation by Global Convolutional Network. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[65] F. Perazzi, J. Pont-Tuset, B. McWilliams, L. V. Gool, M. Gross, and A. Sorkine-Hornung. A Benchmark Dataset and Evaluation Methodology for Video Object Segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[66] J. Pont-Tuset, F. Perazzi, S. Caelles, P. Arbeláez, A. Sorkine-Hornung, and L. Van Gool. The 2017 DAVIS Challenge on Video Object Segmentation. *arXiv:1704.00675 [cs.CV]*, 2017.

[67] E. Romera, J. M. Alvarez, L. M. Bergasa, and R. Arroyo. ERFNet: Efficient Residual Factorized ConvNet for Real-Time Semantic Segmentation. *IEEE Transactions on Intelligent Transportation Systems*, 2018.

[68] O. Ronneberger, P. Fischer, and T. Brox. U-net: Convolutional Networks for Biomedical Image Segmentation. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 2015.

[69] S. B. Sells and R. S. Fixott. Evaluation of Research on Effects of Visual Training on Visual Functions. *American Journal of Ophthalmology*, 44:230–236, 1957.

[70] X. Shi, Z. Chen, H. Wang, D.-Y. Yeung, W.-k. Wong, and W.-c. Woo. Convolutional LSTM Network: A Machine Learning Approach for Precipitation Nowcasting. In *Advances in Neural Information Processing Systems (NIPS)*. 2015.

[71] M. Siam, M. Gamal, M. Abdel-Razek, S. Yogamani, M. Jagersand, and H. Zhang. A Comparative study of Real-time Semantic Segmentation for Autonomous Driving. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops*, 2018.

[72] M. Siam, S. Valipour, M. Jagersand, and N. Ray. Convolutional Gated Recurrent Networks for Video Segmentation. In *Proceedings of the International Conference on Image Processing (ICIP)*, 2018.

[73] D. Silva. Github Page of PyTorch ENet. https://github.com/davidtvs/PyTorch-ENet. Accessed on 2019-05-14., 2017.

[74] K. Simonyan and A. Zisserman. Two-Stream Convolutional Networks for Action Recognition in Videos. In *Advances in Neural Information Processing Systems (NIPS)*. 2014.

[75] K. Simonyan and A. Zisserman. Very Deep Convolutional Networks for Large-Scale Image Recognition. *arXiv:1409.1556 [cs.CV]*, 2014.

[76] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Dropout : A Simple Way to Prevent Neural Networks from Overfitting. *Journal of Machine Learning Research*, 2014.

[77] R. K. Srivastava, K. Greff, and J. Schmidhuber. Highway Networks. *arXiv:1505.00387 [cs.LG]*, 2015.

[78] J. Stallkamp, M. Schlipsing, J. Salmen, and C. Igel. The German Traffic Sign Recognition Benchmark: A Multi-Class Classification Competition. In *Proceedings of the International Joint Conference on Neural Networks*, 2011.

[79] N. Sundaram, T. Brox, and K. Keutzer. Dense Point Trajectories by GPU-Accelerated Large Displacement Optical Flow. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 2010.

[80] I. Sutskever. *Training Recurrent Neural Networks*. Phd thesis, University of Toronto, 2013.

[81] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going Deeper With Convolutions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015.

[82] P. Tokmakov, K. Alahari, and C. Schmid. Learning Video Object Segmentation with Visual Memory. In *Proceedings of the IEEE International Conference on Computer Vision (CVPR)*, 2017.

[83] J. Tompson, R. Goroshin, A. Jain, Y. LeCun, and C. Bregler. Efficient Object Localization using Convolutional Networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015.

[84] S. Valipour, M. Siam, M. Jagersand, and N. Ray. Recurrent Fully Convolutional Networks for Video Segmentation. In *Proceedings of the IEEE Winter Conference on Applications of Computer Vision (WACV)*, 2017.

[85] P. Voigtlaender and B. Leibe. Online Adaptation of Convolutional Neural Networks for Video Object Segmentation. *arXiv:1706.09364 [cs.CV]*, 2017.

[86] W. Yin, K. Kann, M. Yu, and H. Schütze. Comparative Study of CNN and RNN for Natural Language Processing. *arXiv:1702.01923 [cs.CL]*, 2017.

[87] F. Yu and V. Koltun. Multi-Scale Context Aggregation by Dilated Convolutions. *arXiv:1511.07122 [cs.CV]*, 2015.

[88] H. Zhao, J. Shi, X. Qi, X. Wang, and J. Jia. Pyramid Scene Parsing Network. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[89] S. Zheng, S. Jayasumana, B. Romera-Paredes, V. Vineet, Z. Su, D. Du, C. Huang, and P. H. Torr. Conditional random fields as recurrent neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015.

Master's thesis filing card

Student: Johannes Verherstraeten

Title: Enhancing Temporal Consistency in Real-Time Semantic Video Segmentation, Using Recurrent Neural Networks.

UDC: 621.3

Abstract:

State-of-the-art semantic image segmentation methods which are used in autonomous cars, do not make use of temporal relations between consecutive video frames. Each frame is segmented independently from the previous ones. This may result in flickering when small or narrow objects like poles alternately are and are not detected because of small noise influences. Recurrent neural networks may, unlike feed-forward neural networks, take advantage of the temporal information between consecutive frames to enhance the consistency of semantic segmentation.

More specifically, this thesis first explores some recurrent neural network architectures. By making use of this extra temporal information, they should be able to learn that objects don't continuously appear and disappear in video footage. The effect of different recurrent blocks is tested, together with some hyperparameter variations of these recurrent blocks.

The second aspect is the search for appropriate loss functions. By designing loss functions that punish flickering or favour consistency, we can explicitely encourage these networks to learn temporal consistency relations. Next to the warping loss, this thesis proposes two new loss functions: the change and smoothing loss.

The third aspect is the choice of evaluation metrics. This thesis proposes the temporal Intersection over Union metric (temporal IoU), to assess the temporal consistency of consecutive segmentations.

The final aspect is the creation of a framework that allows to train recurrent neural networks on video files. This framework will be made publicly available when ready.

This thesis achieves an increase of 1.91% on accuracy and an increase of 9.43% on temporal consistency with respect to the baseline network, by adding a convolutional recurrent block and a temporal consistency loss. This is achieved with a loss of execution speed of only 2.9%.

Thesis submitted for the degree of Master of Science in Engineering: Computer Science, option Artificial Intelligence

Thesis supervisor: Prof. dr. M.-F. Moens

Assessors: Dr. ir. B. De Brabandere
Dr. ir. G. C. Talleda

Mentors: Dr. ir. M. Proesmans
Ir. J. Heylen