UKRAINIAN CATHOLIC UNIVERSITY

MASTER'S THESIS

Semi-supervised feature sharing for efficient video segmentation

Author:
Anton PONOMARCHUK

Supervisor:
Andrey LUZAN

A thesis submitted in fulfillment of the requirements for the degree of Master of Science

in the

Department of Computer Sciences
Faculty of Applied Sciences

Lviv 2019

UKRAINIAN CATHOLIC UNIVERSITY

Abstract

Faculty of Applied Sciences

Master of Science

Semi-supervised feature sharing for efficient video segmentation

by Anton PONOMARCHUK

In robot sensing and automotive driving domains, producing precise semantic segmentation masks for images can help greatly with environment understanding and, as a result, better interaction with it. These tasks usually need to be processed for images with more the 2 object's classes. Moreover, semantic segmentation should be done for a short period. Almost all approaches that try to solve this task used heavy-weight end-to-end deep neural network or external blocks like GRU [14], LSTM[25] or optical flow [1]. In this work, we provide a deep neural network architecture for learning to extract global high-level features and propagate them among the images that describe the same video's scene, for speeding up image processing. We provide a propagation strategy without any external blocks. We also provide loss function for training such network with the dataset, where the vast number of images don't have a segmentation mask.

Acknowledgements

I would like to thank the supervisor Andrey Luzan for the excellent cooperation and all of the opportunities I was given to conduct this research. Also, I would like to thank Orest Kupyn for the coordination between the supervisor and me.

I would like to thank Oleksii Molchanovskyi and Rostyslav Hryniv for an opportunity to become a part of this master's program and for their valuable guidance.

I would also like to thank Ciklum for providing the scholarship to graduate from this master's program and computational powers for experiments. In addition, I would like to thank Igor Krashenyi for helping to use the Ciklum's computational powers properly.

Contents

Chapter 1

Introduction

Scene understanding by parsing via semantic segmentation is a fundamental topic in computer vision. The main goal is to predict for each pixel in the image a category from a predefined set of classes. Scene understanding predicts the label, location, and shape of each element at the image. Figure 1.1 illustrates examples of input and output images for this task. As input, we take a regular image and produce a semantic mask for each object at the image.

FIGURE 1.1: An illustration of the scene semantic segmentation task. In the input we take some scene (on the left), and produce the output semantic segmentation mask for a scene (on the right). The images are taken from Cityscapes [6] and ADE20K [30] datasets.

The potential applications of this topic are:

- medical image diagnosis - producing precise semantic segmentation masks for lungs, brains, etc. can provide more precise details for doctors and save them time;
- robot sensing - knowing object's precise location and size will help robots to better interact with an environment;
- automatic driving - seeing an object's location size and shape will help cars faster and better understand the environment and ,as a result, make an improved decision.

For automotive driving and robot sensing the task is to provide the semantic segmentation masks for images that have more than two classes. For instance, Figure 1.1 provides image from the street. As we can see, there at least six classes that need to be predicted: nature (trees and grass), the sky, car, road, traffic signs and sidewalk. Producing masks for big number of objects provide difficulties in processing such images. The more classes that need to be processed in the image, the more difficult to do so. Difficulties arise from the variety of classes and sizes of objects, and in different images from the same dataset, objects can be different sizes, which means that we need to enable our algorithm to predict differences in size.

State-of-the-art scene parsing approaches are mostly based on Fully convolutional network (FCN) [20]. These *deep convolutional neural networks*(CNN) work as trainable filters and can adapt to the images need to be processed. The highest result is provided by the architecture DeepLabV3+ [18] with a *mean intersection over union* (mIoU) score 82.1 %.

The main disadvantages of the DeepLabV3+ are the weight and image processing. It has 43.48M [5] parameters that need to be computed during image processin, which causes difficulties when processing a vast number of images during a short period. For instance, during automatic driving, stream-like images need to be processed in a short period. These images are dependent on time, so they have temporal continuity property. As a result, the problem of producing segmentation masks for images for automatic driving can be resolved by video semantic segmentation approaches.

The difference between image and video semantic segmentation is that the approaches for image semantic segmentation process image by image. These approaches process images independently and in general don't use any external information. On the other hand, the approaches for video semantic segmentation try to exploit temporal continuity - by using dependency between frames that describe the same scene. Figure 1.2 illustrates frames that describe one scene. The images contain identical objects with distortion of size and position. The image processing can be speed up and the accuracy can be increased by exploiting this video's property.

FIGURE 1.2: An illustration of video frames that describe same scene. The images are taken from extended Cityscapes dataset.

By using the temporal continuity property, the work [28] tries to build modules for efficient propagation high-level features among high-correlated frames. For using temporal continuity, the video semantic segmentation approaches also use different external elements as a gated recurrent unit [14], long short term memory [25] and optical flow [1]. These elements propagate and adapt the information (it can be feature maps or semantic segmented masks) from processing previous images to the

current one. These elements provide additional time costs. So, the latency for the processing one frame increased.

From [28] we noticed that propagating features from the previous frame to the current one can be done without any external block. Moreover, they used a built-in CNN to adapt feature maps extracted from the previous frame to the current one. In this work, we tried to build such architecture for using extracted high-level features maps from earlier frames to the current one without any implicit or explicit networks or additional elements. To do so, we developed an approach for extracting global high-level features, describing common information about a video scene among high-correlated frames.

We rely on the property that frames describe one scene into a video - contain same global information about the scene that can be propagated. To adapt CNN to extract these features and spread it among frames without any external or internal blocks CNN, is our goal in this paper.

During this work we discovered:

- high-level information among frames sharing strategy - *skip propagation connection* S^{prop};
- loss function for training network with S^{prop} connection and small number of labeled training data in relation to the whole dataset;
- transformed FPN [19] version for semantic segmentation for using S^{prop} and provide results of the benefits and drawbacks by using our approach.

Chapter 2

Related work

After introducing [17] as the state-of-the-art algorithm for image classification problem in 2012, deep learning methods have shown dominant results in various computer vision tasks. For image recognition, classification, and segmentation, deep learning architectures are the major field of research nowadays.

2.1 Image semantic segmentation

The goal for semantic segmentation is to predict per-pixel label in the input image. Great results for this task were achieved by Fully Convolution Networks (FCN). [20] build the end-to-end network that takes an arbitrary sized input image and produce an output of the same size. It was a novel approach for per pixel image segmentation task. The main idea was next: the authors took a convolutional neural network for the image classification problem and the fully-connected layers replaced by convolutional layers, making it possible to get as the output the segmented masks of the input images.

In fact, due to the network structure, this was the first attempt to made a trainable filter, that can be applied to any image size, and the result should be the proportionally sized mask. The authors also developed the approach that achieves good accuracy in an end-to-end way without any additional post-processing like conditional random fields. Moreover, this article has introduced the new direction for developing segmentation approaches. As a result, after that were developed the number of architectures that based on the idea of FCN [20] achieved even better results: U-net [24], Feature pyramid network (FPN) [19], Pyramid Scene Parsing (PSP) [29], [18], see Figure 2.1(the image is taken from [28], with update of DeepLab accuracy from [18]).

Quite often, scenes of the world include elements of the different sizes. These elements contain features of different sizes, too. It is natural to try to develop a system that can process an image at different scales. The image pyramid data structure [10] was developed for efficient scaled image analysis. It consists of a sequence of copies of an original image in which both of sample density and resolution are decreased in regular steps [10].

The goal of [20] is to leverage naturally the pyramidal shape of a ConvNet's feature hierarchy while creating a feature pyramid that has strong semantics at all scales [20]. In other words, [20] provides the neural network with feature pyramids that can replace image pyramids without losing speed, memory capacity, representational power and produces the outputs in a pyramidal manner (the same output but in different scales).

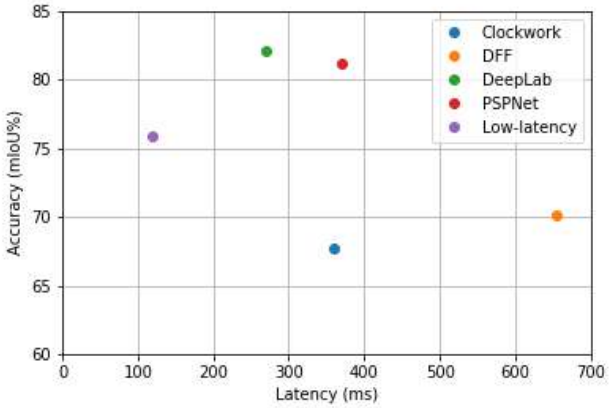

FIGURE 2.1: Comparison of the latency and mIoU on Cityscapes [6] dataset. The architectures are used: PSP [29], Clockwork [27], DFF [31], DeepLabV3+ [18], Low-latency [28].

FPN [19] takes into the input a single-scale image of arbitrary size and produces a proportionally sized feature maps at multiple levels. The network consists of the following elements: *bottom-up pathway*, *top-down pathway*, and *lateral connections*. Figure 2.2 (taken from [19]) provides the FPN architecture in detail.

Bottom-up pathway. The bottom-up pathway is a feed-forward computation of the backbone convolutional network, which consists of *stages*. The stage is a data structure that consists of the layers producing the same size feature maps. It means that, in a stage, several layers can produce the output maps with the same size. One stage corresponds to one level at the network pyramid structure. The outputs of the last layer of each stage are used for creating a pyramid structure. The scaling step between the pyramid's level is two. For instance, in ResNet [13] the feature activation outputs by each stage's last residual block create the pyramid structure. We will denote these outputs as C_2, C_3, C_4, C_5, that are conv2, conv3, conv4, conv5 ResNet [13].

Top-down pathway and lateral connections. The top-down pathway uses feature maps C_2, C_3, C_4, C_5 introduced in a pyramid during bottom-up pathway and creates the final set of feature maps P_1, P_2, P_3, P_4. Firstly, from the last pyramid level (C_5) come the coarser-resolution feature map. Then it is upsampled the spatial resolution by factor 2. After that, the upsampled feature map is merged with the corresponding bottom-up map by element-wise addition. Before the addition, the bottom-up map is proceeded by 1×1 convolution in lateral connection to make the same the number of filters with the top-down map. The processes of upsampling and element-wise addition are iterated until the last (finest) resolution map is generated. To get the predicted output pyramid's feature maps, for each feature map of the top-down pyramid is used 3×3 convolution without any non-linear function after it. As a result, the final set of feature maps is denoted as P_1, P_2, P_3, P_4 which corresponds to C_2, C_3, C_4, C_5.

The state of the art approach of image segmentation is FCN-like neural network - DeepLabV3+ [18]. The accuracy of DeepLabV3+ is 82.1% on the Cityscapes dataset [6].

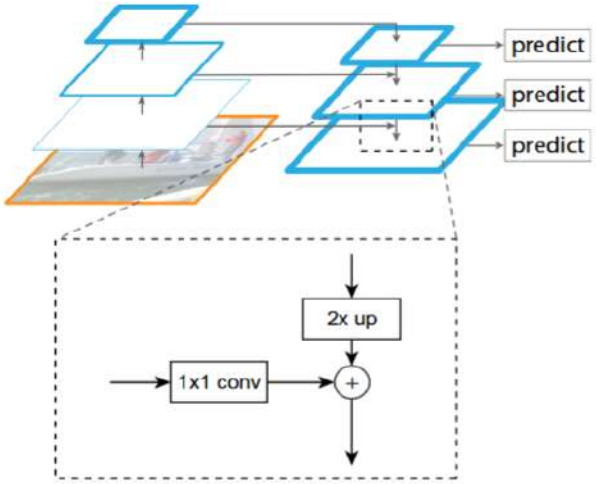

FIGURE 2.2: The architecture of FPN [19]. The bottom-up pathway is the left part of the network. The top-down pathway is the right part of the network. The information from bottom-up to top-down is propagated by lateral connections - arrows that connect both network's parts. Lateral connection contains 1×1 convolutions to make equal the filters number at the feature maps from both parts.

2.2 Video semantic segmentation

Video segmentation is sufficient not only to produce masks with high accuracy, but to do so fast and with as small a latency as possible. FCN-like networks also don't use video's temporal continuity. Specific subsets of images share the same global information, and this information can potentially be reused between frames for reducing the numbers and computations cost. The approaches are separated into three groups:

- using the temporal continuity for increasing masks accuracy;
- decreasing the latency for speeding up image processing;
- combining both of the properties mentioned above.

Architectures based on variations of recurrent neural networks (RNN), optical flow [1], combining both of them, and alternative approaches, were developed to use a video's temporal continuity.

2.2.1 Optical flow approaches

One way to share global information among frames is to use optical flow. Optical flow is a vector field representation of the apparent motion of each pixel between two images. In the case of semantic segmentation, optical flow helps to adapt global information taken from the previous frame for using it during current processing. This means that we can precisely allocate and more efficiently use global information extracted from the previous frame to current.

The optical flow is used at the [7], [31], [23], and [26].

[26] provided end-to-end trainable, highly modular architecture for video segmentation. This system contains three parts that can be chosen and pre-trained independently and fine-tuned as an end-to-end network:

- feature sub-network N_{feat} - takes as input RGB-like image $I_i \in R^{1\times 3\times h\times w}$, where h, w - image height and width respectively. As the result, the subnetwork provides intermediate representation $f_i \in R^{1\times 2048\times \frac{h}{16}\times \frac{w}{16}}$;
- task sub-network N_{task}- takes as input the intermediate representation f_i and returns a semantic segmentation score map $s_i \in R^{1\times C\times h\times w}$, where C - is the number of label classes;
- output block P - takes as input s_i and provides normalized probabilities $p_i \in [0,1]^{1\times C\times h\times w}$. As the result, from p_i can be extracted the segmentation mask $S_i \in R^{1\times 1\times h\times w}$.

Figure 2.3 (taken from [26]) provides the architecture of the [26] in detail. I_k is the key frame. The prediction of the mask S_k for key frame is done through the **reference network**: $S_k = P(N^R_{task}(N^R_{feat}(I_k)))$. For the key frame the computations are done through N^R_{feat} and N^R_{task}. The authors called N^R_{feat} as the **reference feature network**. Usually N^R_{feat} is expensive to compute and deep. For instance, at the paper [26] as it is used a variant of ResNet [13] architecture called Deformable ResNet-101 [8]. This Accel's part extracts intermediates features f^c. Then, f^c is passed to the N_{task} for getting segmentation score map s_k. As the result, the score map s_k is applied the output block P for producing segmentation mask for key frame S_k: $S_k = P(s_k)$. Finally, the output block P contains softmax function, followed by an argmax layer.

Also, the intermediate features f^c extracted from the key frame are cached and propagated through the next (current) frames. For the current frame, a segmentation mask is computed by score maps s^R_i and s^U_i along reference and update branches respectively.

Reference branch computes score map $s^R_i = N_{task}(W(f^c, O(I_{i-1}, I_i)))$, where N_{task} - the task network, and W - wrap the cached features f^c by optical flow field O between the current frame I_i and the previous I_{i-1}. Score map s^U_i also computes by **update network** N^U: $s^U_i = N^U(I_i) = N_{task}(N^U_{feat}(I_i))$, where N^U_{feat} - update feature network. Usually it is used the network that is less deep and expensive then N^R_{feat}. The authors from the article use Deformable ResNet-18, -34, -50. -101 (the bigger the distance from key frame, the deeper and more expensive the network is used for feature extracting from current frame I_i).

After computing score maps s^R_i and s^U_i for prediction segmentation mask S_i they merged by a 1×1 convolution which is called score fusion (SF). The SF produces score map $s_i \in R^{1\times C\times h\times w}$ from stacked score maps $[s^R_i, s^U_i] \in R^{1\times 2C\times h\times w}$. The segmentation mask S_i is computed: $S_i = P(SF([s^R_i, s^U_i]))$, where $[s^R_i, s^U_i]$ - s^R_i and s^U_i are stacked along the channel dimension.

Besides this, it is crucial to note that N^U_{feat}, N^R_{feat}, N_{task} are a separated network that don't share any weights, which means that different deep network architectures can be used, and pre-trained by the way the user needs. The property of modularity gives an opportunity to use diverse deep network architectures, like ResNet modifications, Inception-like [4], DensNet-like [11].

The main disadvantage of the optical flow usage is that the computation costs cannot be reduced due to adding per-frame feature computation.

2.2.2 RNN approaches

Another way to use temporal information for increasing semantic segmentation accuracy is to use different RNN networks like long short term memory (LSTM) [25] networks and gated recurrent neural networks (GRN) [14], as modules.

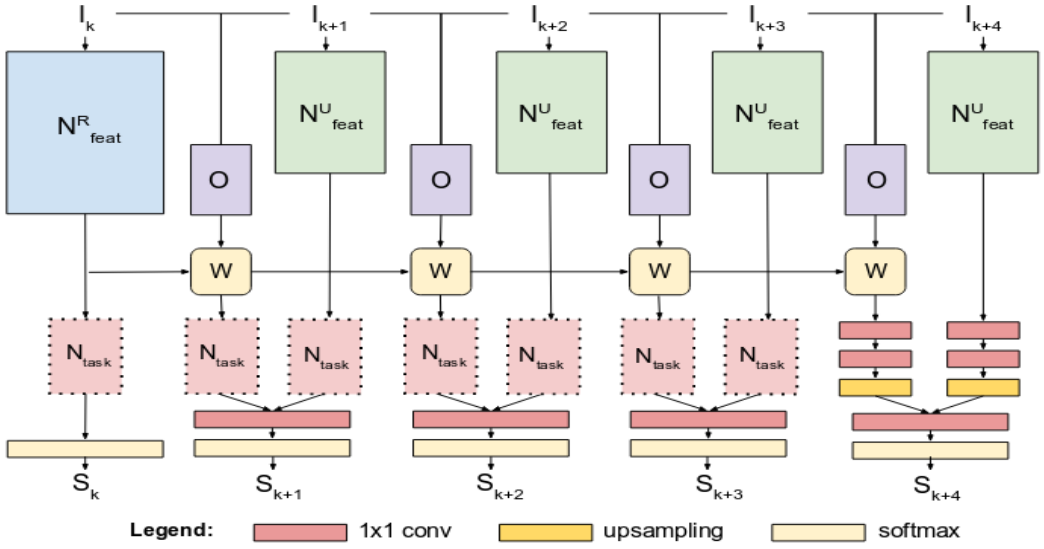

FIGURE 2.3: Accel network [26] architecture. N^R_{feat} - reference feature sub-network, can be different ResNet architecture depending on distance between key frame, and others; N_{task} - task sub-network; O - optical flow field; W - wrapper for features computed by optical flow field O.

A recurrent neural network (RNN) is an extension of a conventional feedforward neural network, which can handle a variable-length sequence input. [14] The RNN handles the variable-length sequence by having a recurrent hidden state whose activation at each time is dependent on that of the previous time. [14]

A long short term memory (LSTM) network is a type of recurrent neural network that has been purposed in [25] for remembering information during long periods and solving the vanishing gradient problem. LSTM can decide whether and how long to keep necessary features from an input at the early stage.

A gated recurrent network (GRN)[15] is a type of recurrent neural network that can adaptively capture dependencies of different time scales. It is similar to LSTM by properties but has some differences in implementation of the gating units.

The LSTM and GRU units are used in [7], [21], and [2] remembering features from previous frames and reusing them for producing semantic segmentation masks.

There also exists research that tries to combine RNN-like modules with optical flow. In [7] the GRU units with the optical flow are used for better adaptation global information and increasing output results. The optical flow is used for warping the semantic segmentation estimates from previous frames. Moreover, GRU is used for connecting the warped representation of the previous image segmentation mask with the current probability candidate mask. More information about this approach is described below.

In the article *Semantic video segmentation by gated recurrent flow propagation* [7] tries to exploit the temporal continuity of the images at the video for increasing accuracy. In this article, the authors presented the *spatio-temporal transformer gated recurrent unit* (STGRU). This unit composes the information from the previous image mask was warped by optical flow and current image segmentation mask was created by a convolutional neural network. It consists of:

- optical flow function ϕ;

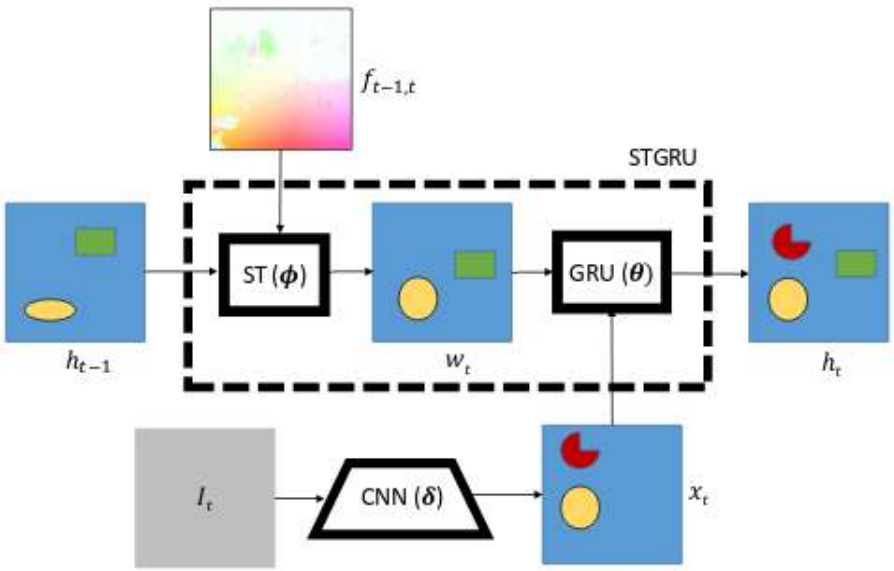

FIGURE 2.4: Architecture for model with gated recurrent flow propagation [7]. The semantic segmentation mask x_t for image I_t computed by convolutional network δ. It is used in GRU module θ with previous image segmentation mask h_{t-1} wrapped along optical flow ϕ for predicting output semantic segmentation mask h_t for image I_t. The optical flow function ϕ and GRU module θ create STGRU module.

- gated recurrent unit (GRU) [14];
- CNN for image segmentation.

The main idea is the next. Look at Figure 2.4 (taken from [7]). The segmented mask from previous image h_{t-1} is warped along the optical flow to align with the segmentation mask in time t. This is done by computing $w_t = \phi_{t-t,t}(h_t - 1)$, $\phi_{t-1,t}$ is the mapping along optical flow. The w_t is a hidden state at the GRU, which get current frame segmentation mask x_t as the input. The x_t is computed by convolutional neural network over current frame I_t. After that, the GRU calculates the final segmentation mask h_t for frame t using information from x_t and w_t.

2.2.3 Alternetive approaches

Despite of the attempts to use temporal continuity for increasing image segmentation maps accuracy, there exist alternative approaches that focus on reducing the computation cost: [28], [27], and [31].

Our work was inspired by "Low-latency for semantic segmentation" paper [28]. This paper introduced a video segmentation approach that has low latency (119 ms), reuses features from the previous frames and produces the competitive accuracy with state of the art approaches (nearly 76% mIOU).

The authors have proposed a framework for segmentation where the part of the information from key-frames is propagated to the others by exploiting the high correlation between adjacent frames. The key-frames are images that cannot be segmented by using additional information from previous ones because dramatically different from previous ones. These frames should be computed through the whole pipeline, and the part of the features will be propagated to the next adjacent images. On the other hand, the images that are highly correlated with key-frames can reuse

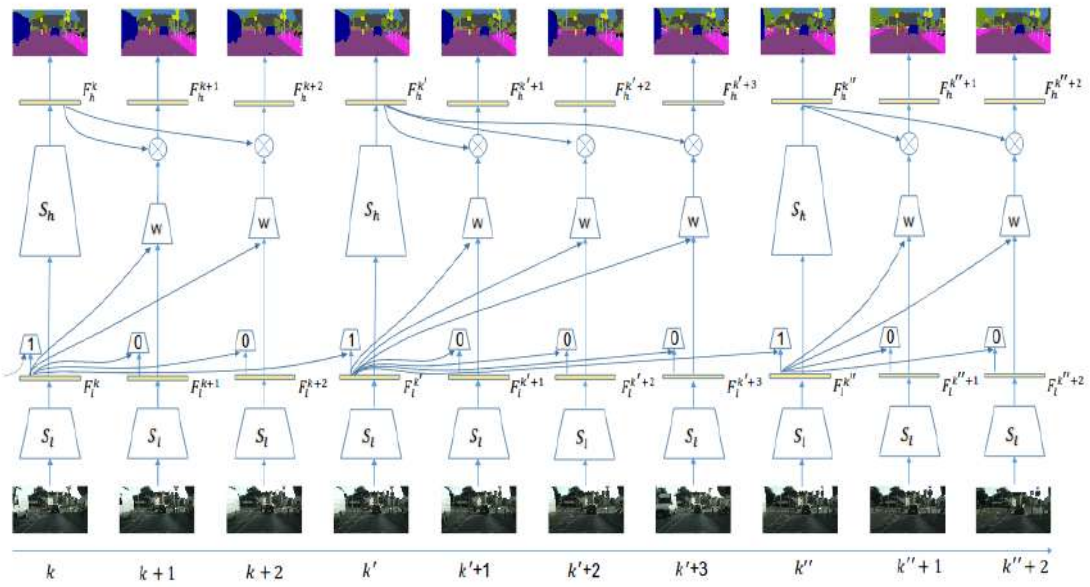

FIGURE 2.5: Low-latency approach described in [28]. S_l - lower-part, extracts low-level features from every frame; S_h - higher-part, extract high-level features from key-frames that used for propagation through current frames; w - kernel weight predictor; $0, 1$ - network that decides if the current frame is to far from key-frame or not.

a part of features (information) from key-frames. These images are called current frames.

The framework extracts two types of features from the image: low-level and high-level. The low-level features are the information obtained from lower layers of a CNN. They are cheap to compute but contain useful information for creating segmentation masks. This type of feature is processed for all the frames. They are also used for detecting whether the current frame is crucial. On the other hand, high-level information is costly to compute. This information is processed only for key-frames and reused by the next frames that describe the same scene, the "Low-latency" approach covers solutions for two tasks: the key-frame selection and the propagation of features across frames.

The network has two parts:

- lower-part S_l - from this part low-level features for each image are derived. These features are used for selecting key-frames and controlling high-level features propagation.
- higher-part S_h - from this part high-level features for key-frames are derived. These features are propagated from key-frames to others, that are correlated with key-frame.

Figure 2.5 (taken from [28]) represents the whole pipeline. Firstly, $t - th$ frame I_t the S_l part computes the low-level features F_l^t. After that, the additional small convolutional neural network based on F_l^t decides whether frame I_t is key. If the result is positive, then F_l^t feed to the high-part S_h and calculate high-level features F_h^t. Otherwise, F_l^t is fused with key-frame low-level features F_l^k by a small network. They called this network *kernel weighted predictor*, which takes as input both F_l^k and F_l^t. It produces the kernels at the all spatial locations. The result from the kernel

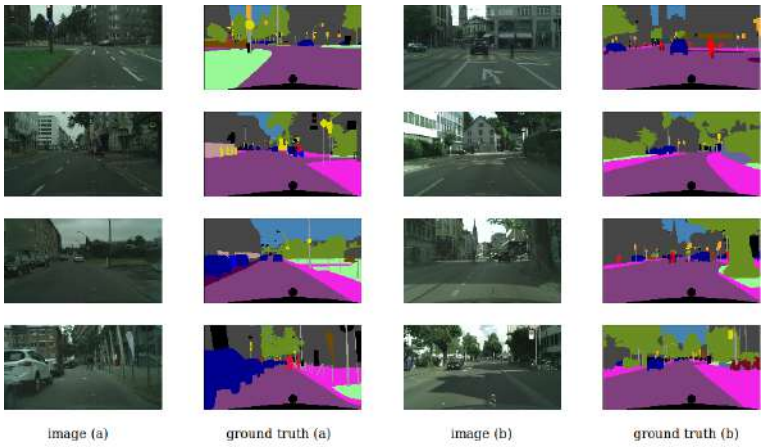

FIGURE 2.6: Examples of images and corresponding ground truth mask from train set at the Cityscapes.

weighted predictor is used for propagating the high-level features from the previous key-frame by spatially variant convolution.

For propagating global features, [28] uses *spatially variant convolution*. The transformation global features from key frame F^h to current F^t:

$$F_h^t(l,i,j) = \sum_{u=-\Delta}^{\Delta} \sum_{v=-\Delta}^{\Delta} W_{ij}^{k,t}(u,v) \times F_h^k(l, i-u, j-v)$$

where $\Delta = [\frac{H_K}{2}]$, $F_h^t(l,i,j)$ is the feature value at (i,j) of the l-th channel in F_h^t. $W_{ij}^{k,t}$ if the $H \times H$ kernel used for computing the feature at i,j position during propagation from F_k^h to F_t^h. [28].

For predicting kernel $W^{k,t}$ values, [28] uses a *kernel weight predictor* that is able to predict the kernels for all location. It is a neural network that takes both low-level features F_k^h and high F_t^h to produce the set of kernels.

2.3 Dataset

For the training baseline and our approach is used Cityscapes [6] dataset. Cityscapes is a dataset for semantic understandings of urban street scenes [6]. The *conventional* version consists of 5000 high quality (1024×2048), finely annotated images from 50 cities during several seasons (spring, summer, and fall). Figure 2.6 illustrates examples of the images and correspondent segmentation masks. The images are divided into *train*, *validation*, and *test* sets with 2975, 500, and 1525 images respectively. The dataset defines 19 categories. The authors have divided each scene into 30 images and segmented each 19-th frame out of 30. It means that each frame in this dataset represents a unique scene in the video.

This information is not enough for training our approach because we need to extract the information from images that high-correlated with already segmented ones. On the other words, for this purpose, we need more images from each scene, and these images should be previously to the already annotated ones. For this purpose, we requested an extended version of this dataset. By *extended* version means the videos divided into 30 frames per scene. Each 19-th frame out of 30 has the ground truth segmentation mask from the conventional dataset.

Chapter 3

Video segmentation approach

There exists a video I composed of the set of frames I_{set}: $\{I_1, I_2, ..., I_n\}$, where $n \in \overline{[1, N]}$, N is the total number of frames in the video I. The goal is for any frame $I_n \in I_{set}$ to compute the semantic segmentation mask $S_n \in S_{set}$. It means finding such a mapping

$$P : I_{set} \rightarrow S_{set}$$

As has already been mentioned in chapter 2.1, it can be done by deep neural networks for image semantic segmentation like PSP [29] or DeepLabV3+[18]. They are accurate but slow, see Figure 2.1 (taken from [28] and [18]). Furthermore, the algorithms don't use the spatial continuity information because the networks compute frames independently during forward pass.

Again from chapter 2.1, the video has a temporal continuity property. The efficient usage of this property can bring the advantage of creating high-quality semantic segmentation mask and decreasing the total latency. Whereas the video has temporal continuity property, it means that a sub-sequence of frames $I_{scene} \subset I_{set}$ contains the same high-level information F^{high} about the video scene.

This chapter describes the way to propagate high-level features F^{high} through highly correlated frames in the video. It also explains the architecture of the deep neural network used for efficient feature propagation and semantic video segmentation.

Our approach is adopted from the paradigm of propagating high-level information F^{high} from the key frame $I^{key} \in I_{set}$ to the highly correlated current frame $I^{current} \in I_{set}$, by exploiting the high correlation between adjustment frames as described in [27], [31], and [28].

The key-frame I^{key} cannot be segmented by using high-level features F^{high} extracted from previous images at the video. These frames contain information about a new scene at the video and low correlated with the last key-frame.

On the other hand, the current frame $I^{current}$ is an image that can be segmented by using extracted and cached information F^{high} from the key-frame I^{key}. Often these images describe the same scene as the key-frame I^{key}, and as a result, are highly correlated. The video semantic segmentation is done by frame. For $I_n \in I_{set}$, where $n \in \overline{[1, N]}$, $S_n = P(I_n)$, where $S_n \in S_{set}$.

The approach described below is general for the arbitrary image semantic segmentation network. All the networks described above can be transformed by this approach, which can be used for: FCN [20], U-net [24], PSP [29], FPN [19], DeepLabV3+ [18] etc.

3.1 Model

We took FPN [19] with DenseNet121 [11] backbone as the base for our network and adapted it according to our semantic segmentation approach. As described in 2.1 FPN consists of the four levels. Each level contains the dense block from DenseNet 121 [11]. The DenseNet naturally transforms to the FPN architecture because it has four dense blocks, and as a result, these blocks become levels in our architecture.

The semantic segmentation network P consists of the four elements in Figure 3.1:

- low-level feature extractor N^{lfe};
- high-level feature extractor N^{gfe};
- decoder N^{d};
- skip connection S^{prop}.

Firstly, each frames $I_n \in I_{set}$ are propagated through low-level feature extractor N^{lfe}. N^{lfe} is a network that generates the low-level features F_n^{low}:

$$F_n^{low} = N^{lfe}(I_n)$$

It consists of some amount FPN's levels. The number of levels p_{low} that are used in N^{lfe} depends on the amount of cached information that should be propagated among current frames I_c. This number changes: $p_{low} \in \overline{[1,3]}$.

After that, depending on whether I_n is a key-frame, the high-level features are computed differently. For key-frame I_n^{key} (the n-th frame in video I is a key), the high-level features F^{high} is computed by using a high-level feature extractor N^{gfe}:

$$F_n^{high} = N^{gfe}(F_n^{low})$$

N^{gfe} is a network that generates the high-level features maps. To decrease the latency for producing segmentation masks, N^{gfe} is used only for key-frames I_n^k. Moreover, high-level features extracted from key frames F_n^{high} are cached and propagated through high correlated current frames I_n^c. The propagation is done by skip connection S^{prop}. The detailed process of the feature propagation is described in 3.2. The number of levels p_{high} in N^{gfe} depends on the levels number in N^{lfe}.

Finally, the high-level features F_n^{high} and low-level features F_n^{low} are passed to the decoder N^{d}, which produces the semantic segmentation mask S_n for input frame I_n:

$$S_n = N^{d}(F_n^{high}, F_n^{low})$$

The decoder N^{d} part is used for all frames because it combines high- and low-level features for increasing the quality of the segmentation mask output. The same is done in [20], [24], and [19]. In case of the FPN baseline, the decoder consists of three convolutional layers, two upsample layers, two batch normalization layers and two ReLU activation functions.

For key frame I_k the tensor T_f^k is computed:

$$T_f^k = P^k(I_k) = N^{d}(N^{gfe}(N^{lfe}(I_k)), N^{lfe}(I_k))$$

Also, for current frame I_c, the T_f^c is computed:

$$T_f^c = P^c(I_c) = N^{d}(S^{prop}(I_k), N^{lfe}(I_c)),$$

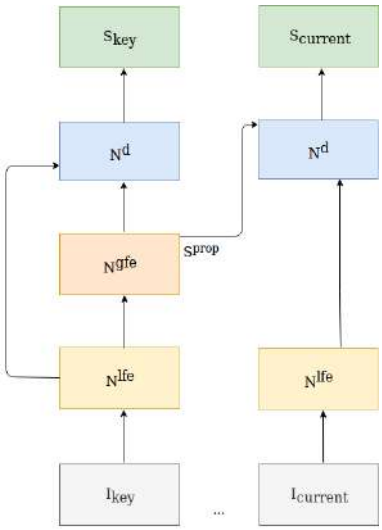

FIGURE 3.1: Semi-supervised approach for efficient video semantic segmentation. It consists of two paths for key-frames (left pipeline) and current-frames (right pipeline). For current-frames, the high features maps are not computing. The are cached during key-frames calculation and propagated by skip connection S^{prop}.

where P^c and P^k are network modes for computing segmentation masks for key and current frames respectively.

The final layer of the network predicts $m \times n \times k$ tensor T_f, where $m \times n$ - $height \times width$, and $k \in K$ - the number of classes to predict. In other words, the approach returns the tensors that consist of the feature maps for each class per pixel. After that, to predict probabilities for each pixel we applied softmax function [12]:

$$softmax(z)_{ij}^k = \frac{exp(z_{ij}^k)}{\sum_l \sum_p exp(z_{lp}^k)}$$

$softmax(z)_{ij}^k$ - "probabilities" to get class k at the position ij at the image, where $i \in \overline{[1,n]}$ and $j \in \overline{[1,m]}$.

$$T_f^{prob} = softmax(T_f)$$

To get a semantic segmentation mask S_l for frame l we apply max function along all the classes:

$$S_l^{ij} = max_{k \in \overline{[1,K]}}(T_f^{prob})_k^{ij}$$

. For each pixel ij we get the most likely class long all classes. As the result $S_l = max_{k \in \overline{[1,K]}}(T_f^{prob})_k$, S_l is the semantic segmentation mask for input frame I_l.

3.2 Feature propagation

One of the neural network properties is the ability to learn the filters for processing images. This property is an advantage over the hand-craft features because the learning algorithm tries to find the optimal values to filters due to available dataset and loss function.

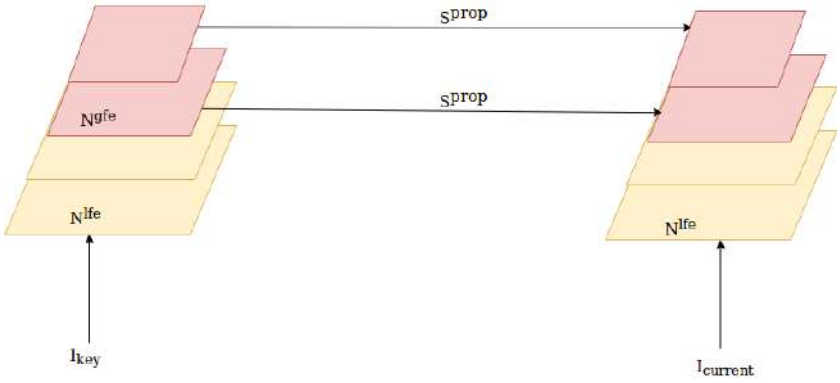

FIGURE 3.2: Skip connection for high level features from key-frame processing to current-frame.

The idea of feature propagation is the next: to adapt the neural network filters to process the key-frames I_k by all them and current ones I_c only with a subset of them. The reduction of the number the active filters causes the latency to decrease. At the same time, to prevent decreasing quality, the information from blocked computations need to be compensated. Since the video has temporal continuity property, missed information can be extracted from key-frames and propagated along current ones. Approaches to propagating this information by using optical flow and spatially variant convolution [28] are described in 2.1. The disadvantage of these approaches is that they need to build on additional blocks for computations. These blocks lead to increasing number of computations.

Our approach is an attempt to propagate information without any additional building blocks for temporal continuity computing. In other words, we tried to teach the neural network to find and use the context that doesn't change through frames and propagate it across highly correlated images. To do so, we freeze a subset of the high-level features F^{high} extracted during key-frame processing I_k and cached. Then, this information is used directly without any adaptive blocks at the processing pipeline for the current image I_c. It was done in a semi-supervised manner because we didn't control the way what network propagate features from key to the current frame.

Moreover, because of the limitations described in chapter 4, we have extracted high-level features for propagating from images without semantic segmentation masks. It caused to forgetting how to extract high-level features for predicting semantic segmentation mask for key-frames. As a result the network start to predict worse, because when our network forget how to extract features correctly from key-frames, then we propagated the wrong features among current frames. The ability of the extracting correct features is critical for our approach.

Figure 3.2 shows the propagation method. Two out four levels are cached and are not computed during current frames processing.

We tried to cache a different number of the FPN's levels, but the more levels we cached, the lower result we got. On the other hand, even for caching half of the levels (two out of four), the accuracy was almost the same with the baselines, see Figure 5.2.

Chapter 4

Loss function and accuracy

For training such a network we need to solve two problems: which loss function to use and how to measure the accuracy of the predicted segmentation mask by our algorithm. For measuring the accuracy we decided to use the standard approach - mean intersection over union (mIoU) or Jaccard index[20]:

$$mIoU = \frac{1}{n_{classes}} \sum_i \frac{n_{ii}}{t_i + \sum_j n_{ji} - n_{ii}}$$

where
$n_{classes}$ - total number classes to predict

n_{ij} - total number of pixels of class i predicted to belong to class j
$t_i = \sum_j n_{ij}$ - the total number of pixels of class i. $mIoU \in [0, 1]$.

Our architecture should create semantic segmentation masks for the current frames by using high-level features F_{key}^{high} extracted from key-frames. Our network should be able to extract high-level features from the images that are invariant for frames from the same video scene. As a result, the loss function should take into account errors between high-level features from key F_{key}^{high} and current $F_{current}^{high}$ frames.

Due to the nature of the data, the vast part of the frames in the extended Cityscapes dataset are without true segmentation masks because the cost for creating multiclass semantic segmentation mask for each frame into a video is too expensive. To deal with this issue, as the current frames, we took the images with true semantic segmentation masks. They are from *train* and *val* parts at the conventional Cityscapes dataset [6]. As the key-frames I_{key}, we took images from the extended dataset. On the other hand, as current frame $I_{current}$ we took an image with semantic segmentation mask from the same scene.

We built the samples for training our architecture in a way of triplets: mask for current image $S_{current}^{true}$, $I_{current}$, I_{key}. For I_{key} we took one out 15 preceding frames from the same scene as the $I_{current}$ randomly as shown in Figure 4.1. This process allowed us to extract high-level features F^{high} from images at the different distances $(I_{t-15}...I_{t-1})$ from image with semantic segmentation mask I_t.

The loss function L_{total}:

$$L_{total} = \sum_{i=1}^{N} \alpha_{dst}^{i} [L_{nll}(y_{current}^{pred}, y_{current}^{true})^i] + \sum_{j=1}^{N} (1 - \alpha_{dst}^{j}) [L_{nll}(y_{key_1}^{pred}, y_{key_1}^{true})^j + L_{nll}(y_{key_2}^{pred}, y_{key_2}^{true})^j] + \\ + \lambda \sum_{k=1}^{K} \|w_{current}^{k} - w_{key}^{k}\|$$

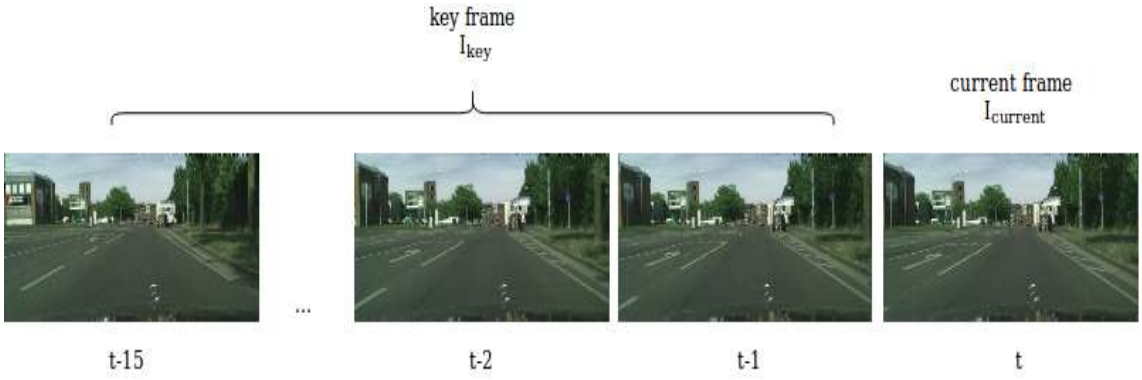

FIGURE 4.1: The approach to choose key and current frames. Each scene consists of an image with segmented mask t and images without segmented masks $t-1...t-15$. The frame with segmentation mask is treated as $I_{current}$ and one out 15 images without segmentation mask as I_{key}. The images are taken from the extended version of the Cityscapes dataset.

The L_{nll} is the negative log likelihood loss:

$$L_{nll} = -log(\sum_{k=1}^{c}\sum_{i=1}^{h}\sum_{j=1}^{w}(y^{pred} \cdot y^{true})_{ij}^{c}),$$

where $y^{pred} \in \mathbb{R}^{h\times w\times c}$ and $y^{true} \in \{0,1\}^{h\times w\times c}$.

The α_{dst} is a weight that regulates importance for the correct prediction in loss function due to the distance between key and current frames:

$$\alpha_{dst} = \frac{1}{d+1},$$

where $d \in \overline{[2,15]}$. The further a key frame is from current one, the less important features can be extracted from it.

As already mentioned, not all frames from the extended dataset have semantic segmentation masks. Only every 19-th frame has been segmented by hands. The goal of our network is not only to be able to extract features from key-frames and use them in creating the mask for current frames but also to be able to create the mask for key-frames. To deal with these tasks, we trained our model in a semi-supervised manner (all key frames during the training process are without true segmentation masks).

To count errors during predicting masks for the key frames, the loss function L_{total} contains:

$$\sum_{j=1}^{N}(1-\alpha_{dst}^{j})[L_{nll}(y_{key_1}^{pred}, y_{key_1}^{true})^{j} + L_{nll}(y_{key_2}^{pred}, y_{key_2}^{true})^{j}],$$

where $N \in \mathbb{R}$ - the total number of samples for training.

We combined two approaches: 1) to augment the quantity of the semantic segmentation masks by other semantic segmentation neural network; 2) to use the current images as the key ones to count losses during key-frames segmentation process.

Increasing the number of masks for frames can be done by the side neural networks. In our training process, the true artificial output key mask $y_{key_1}^{true}$ for key-frame

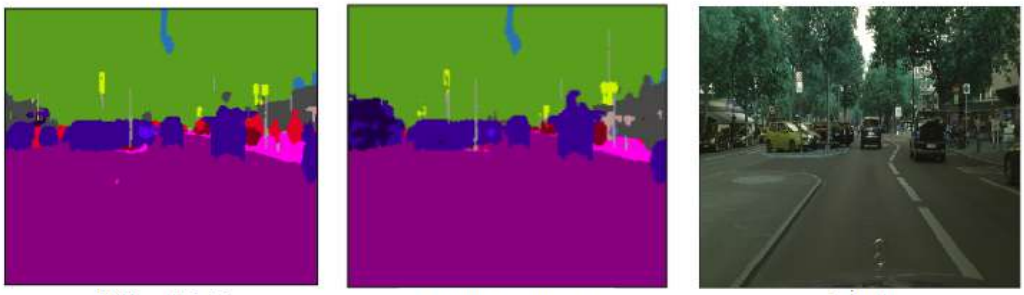

FIGURE 4.2: The difference between training network with and without artificial masks for key frames generated by semantic segmentation network. The image at the left side is more accurate then the right one.

I_{key} is predicted by FPN (under FPN we means adapted to semantic segmentation masks prediction task FPN like network-based by DenseNet-121):

$$y^{true}_{key_1} = P_{FPN}(I_{key}).$$

Moreover, as already mentioned there exists another way to teach the network to segment key-frames. We can treat the current frame as the key, and generate the masks for the current frame by key and current network modes. We computed segmentation mask $y^{pred}_{key_2}$:

$$y^{pred}_{key_2} = P^k(I_{current}).$$

To count errors during predicting masks for current frames, the loss function also contains:

$$\sum_{i=1}^{N} \alpha^i_{dst}[L_{nll}(y^{pred}_{current}, y^{true}_{current})^i],$$

where:

$$y^{pred}_{current} = P^c(I_{current}).$$

To count the loss for the current frame we applied key frame without segmentation mask to extract high-level features and predict segmentation mask for current frame.

Both of these algorithms are done without our direct impact on the learning process of the prediction segmentation masks for the key -frames. In the case with an outside neural network, we couldn't get a true mask for key-frames that exceeds 82% mIoU (DeepLabV3+ has accuracy nearly 82 % mIoU).

Moreover, during training, we figured out that teaching to predict segmentation masks for key-frames even with artificial masks brings benefits to see Figure 4.2. As a result, the predicted mask from our approach (the left one in Figure 4.2) using artificial true masks $y^{true}_{key_1}$ during training process provide more accurate and boundary precise output masks for objects, unlike to the trained network by without artificial masks (right segmentation mask in the Figure 4.2). The usage of the artificial mask means adding $L_{nll}(y^{pred}_{key_1}, y^{true}_{key_1})$ to the loss function.

We have used both approaches to count loss for key frames. The artificial masks are used for teaching how to predict semantic masks for real key-frames. On the

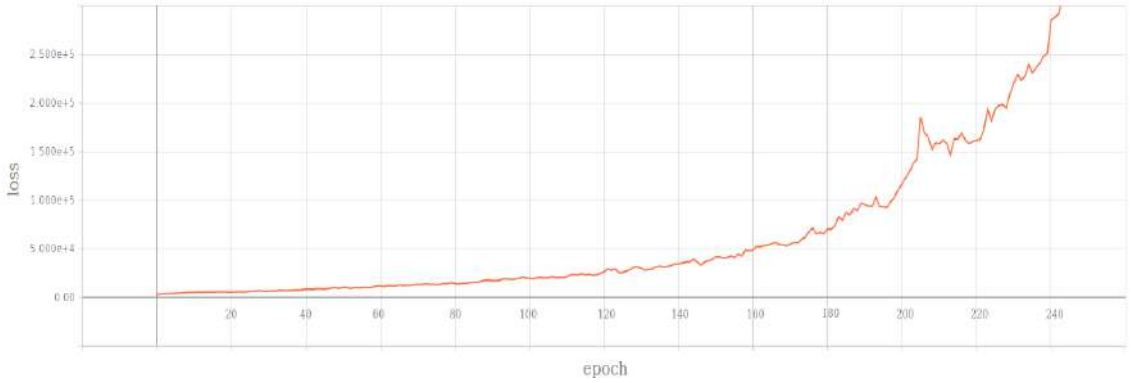

FIGURE 4.3: The loss started grow up after freezing high-feature maps for current images.

other hand, to compensate for the noisy artificial semantic masks (because the accuracy for FPN don't exceed 66.3 % see Table 5.2), we used the second approach with replacing key-frames by current and compute the loss as for key ones.

While training our approach, we noticed that freezing the computation of the high-level feature maps for current frames is not enough to teach recognition of the constant global context that is not changing across the frames from the same scene. As a result, the loss started to grow as in Figure 4.3. The loss started to grow because the network didn't count the difference between high-level features in the current and key-frames. So, the neural network adapted high-level features extraction by key-frames.

To force the algorithm to teach to extract constant high-level information and to prevent the loss growing as in Figure 4.3, we applied Lasso (L_1) regularization [3] between high-level feature maps $F^{high}_{current}$ and F^{high}_{key} from key and current frames:

$$\lambda \sum_{k=1}^{K} \|w^k_{current} - w^k_{key}\|,$$

where $\lambda \in \mathbb{R}$ is a hyperparameter and can be fine-tuned.

We penalized the network when the extracted global context of the key-frame significantly diverged from the current frame. As already mentioned, during the training session we computed a high-level feature maps for current frames $F^{high}_{current}$, but during the validation and evaluation they are not predicted for current frames and propagated from corresponding key-frames.

Chapter 5

Experiments

To demonstrate the advantages of our network, we first trained the baseline. Then we showed that the baseline adaptation up to our algorithm brings benefits by using a video's temporal continuity property. Lastly, we explained that our algorithm also brings benefits under truncated FPN model.

We trained the baseline, our algorithm and truncated model into conventional and extended Cityscapes versions. The conventional one consists of 5000 images: 2975 trains set, 500 validation set and 1525 test set. The extended one also consists of full videos split by frames. The networks were trained by using PyTorch [22] framework.

Baseline. As the baseline we took FPN [19] with pre-trained DenseNet-121[11] as backbone by ImageNet [9] dataset. There can be used other neural networks as backbones; for instance ResNet [13] is used in [19], that describes FPN architecture. Besides, FPN architecture can be used for different purposes: semantic segmentation, object detection, etc. [19]. We used FPN adapted by semantic segmentation problem.

We trained it by conventional Cityscapes dataset. As augmentations, we used *random rotate* to 10 degree and *random horizontal flip*. We also re-sized images from 1024×2048 to 512×512. For optimization we used Adam optimizer [16] with the initial *learning rate (lr)* = 10^{-4} and *weight decay* = 5×10^{-4}, where weight decay is a hyperparamter for L_2 (Ridge) penalty [3] at the optimization process. We additionally applied exponential learning rate policy:

$$lr = lr_{prev} * e^{-k*iteration}, \tag{5.1}$$

where $lr_{prev} \in \mathbb{R}$ - current learning rate, $k \in \mathbb{R}$ - hyperparametr, $k = 0.1$, $iteration \in$

Average latency (ms)		
Distance	1 layer	2 layers
truncated	6.8	6.8
baseline	14.6	14.6
2 frames	12.6	9.38
5 frames	11.6	7
10 frames	11.5	6.93
15 frames	11.4	6.91

TABLE 5.1: Average latency for predicting semantic segmentation masks for different distances between key and current frames and the number of propagating high-feature maps on the extended Cityscapes dataset.

Accuracy (mIoU%)					
Current mIoU	Key mIoU	Cached levels	Reg	α_{dst}	lr
58.5	58.5	truncated (-2)	-	-	expon
66.3	66.3	baseline	-	-	expon
58	60.2	1	L_2	0.3	const
65.25	65.7	1	L_1	float	expon
55	60	2	L_2	0.3	const
56	63	2	L_1	0.3	const
58.6	65	2	L_1	float	expon

TABLE 5.2: The accuracy of the key and current frames by different setups at the *val set* on the extended Cityscapes dataset. *float* α_{dst} means the distance depending α_{dst}. *expon* for lr means exponential learn policy. *-2* cached levels - truncated model, 2 levels were removed. All experiments is done with random training distance between key-current frames.

$\mathbb{R}$ - iteration number. The *updated learning step* = 30 epochs. It means that we applied 5.1 to update lr every 30 epochs, and we used the negative log likelihood loss function As a result, the mIoU for baseline network - FPN at the validation set is 66.72%. It was training for 100 epochs. The average latency is 14.6 ms on the extended Cityscapes dataset.

Truncated model. As the truncated model we took the FPN from the baseline but removed the last two levels from the pyramid structure. It sped up our models but decreased the accuracy of predicting semantic segmentation masks. We have trained it the same as the baseline with the identical setups.

Our approach. We adapted the baseline FPN to our architecture. We tried to propagate different numbers of levels at the FPN as described in chapter 3.2. We propagated features from one and two levels. While training our model, we used a loss function described in chapter 4. In the loss function, the hyperparameter $\lambda = 0.01$ with different norms - L_1 and L_2. We used Adam optimizer with initial $lr = 10^{-4}$ and *weight decay*. We tried with and without exponential learning rate policy 5.1 with $k = 0.1$ and *update learning rate* = 30. Moreover, we tried different weighting approaches for regulating the importance of the correct prediction in loss function due to the distance between frames: constant weighting $\alpha_{dst} = 0.3$; depending on the distance between frame as described in 4. We have trained all our approaches across 75 epochs. We also used baseline as the pre-trained model during training. Besides this, we used random and constant distances for choosing key-current frame pairs. By random distance, we mean that for each pair key-current frames the distance in all samples in training and validation sets are chosen at random. On the other hand, by constant distance, we mean that the distance between key and current frame for all samples are constant. It can be 2, 5 or 10 frames. For simplifying illustrated accuracy-latency dependency, we merged accuracy for key and current frame by weighting sum:

$$mean_{accuray} = \frac{1}{dist} * mIoU_{key} + \frac{dist - 1}{dist} * mIoU_{current}$$

where $mIoU_{key}$ and $mIoU_{current} \in [0, 100]$ - mean intersection over union for key and current frames; $dist \in \{2, 5, 10\}$ - distance that was used during validation evaluating.

The accuracy for all models was evaluated by the validation sets from regular

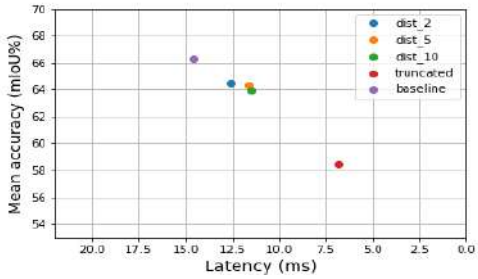

FIGURE 5.1: The comparison of the accuracy and speed between truncated model, baseline and our architectures for constant distances for 2, 5, 10 frames. For our implementations we propagated feature maps from 1 FPN's level. For each distance was trained individual model with corresponding constant distance during training process.

(truncated FPN, baseline) and extended (all our approaches) Cityscape versions. Table 5.2 presented accuracy for truncated FPN, baseline, modification of our algorithm with random training distance. Our implementations slightly worse for then the baseline, especially, with 1 cached level 65.7 % against 67 %. Moreover, the mIoU for current frame was slightly worse for the key-frame during caching one FPN's level. For caching two levels, the accuracy for current and key-frames is slightly different. The more features we need to propagate, the more lower-level features we need to spread. So, the less accurate segmentation mask we would get. As a result, for caching two levels implementation the accuracy for current frames is lower than for key-frame (58.6 % against 65%) and one level implementation (58.6 % against 65.25%).

As already mentioned in chapter 1, we didn't use any algorithm for determining the key and current frames. As a result, we tried for different constant distances between key-frame and current frames: 2 images, 5 images, 10 images, and 15 images (the process of the generating key and current frames pairs is described in chapter 4). We took 500 sequential images from the extended version of the Cityscapes and measured the time for predicting semantic segmentation masks by our approach during different distances between key-frame and last current key. We have measured time for predicting two cases: high-level information propagated through one FPN's level and two FPN's levels.

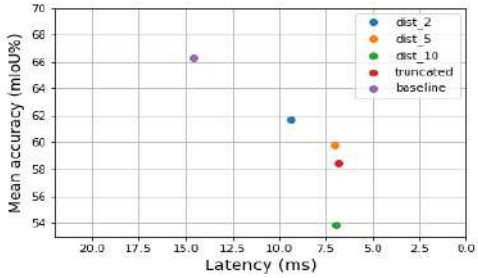

FIGURE 5.2: The comparison of the accuracy and speed between truncated model, baseline and our architectures for constant distances for 2, 5, 10 frames. For our implementations we propagated feature maps from 2 FPN's levels. For each distance was trained individual model with corresponding constant distance during training process.

The results of the latency are shown in Table 5.1. The more levels we have cached,

the faster segmentation mask we would get. Furthermore, the bigger distance between key-frame and last current frame, the faster the network we achieved. For distance two (every second frame is key) we got an average 14 % for one cached level and up to 36 % for two cached levels of the speeding up. There exists a trade-off between accuracy and latency. The more accurate the architecture, the less speeding up would be gained and vice versa. But even for caching one of FPN's level, the speeding up is nearly 14 %, and the accuracy is lower less than 2%.

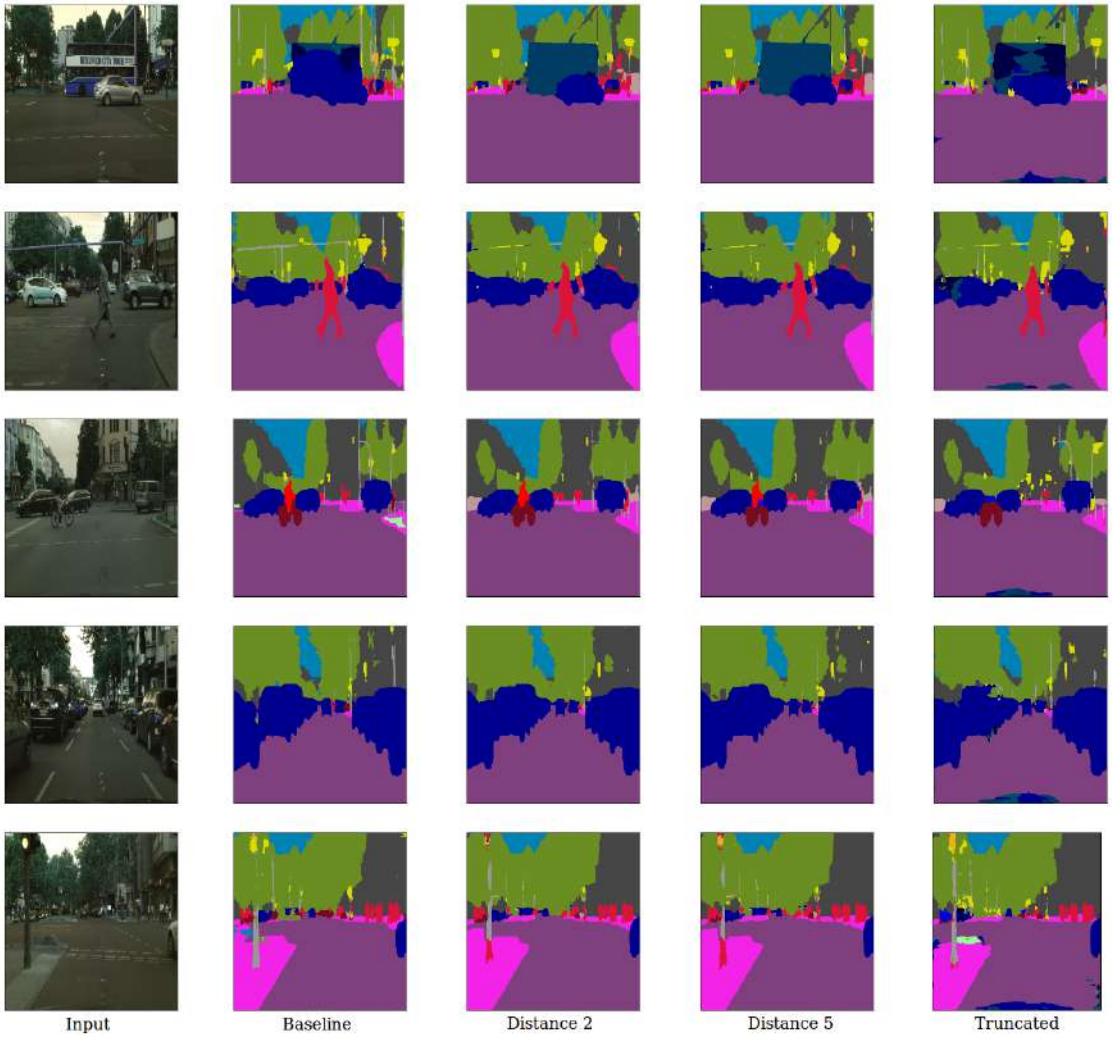

FIGURE 5.3: Examples of images and corresponding predicted semantic segmentation masks by: baseline (FPN); our approach with 2 propagation levels with constant distances 2 and 5 between key and last current frames; truncated 2 levels FPN. The input images are from test part of the extended Cityscapes dataset.

Figures 5.1 and 5.2 show the accuracy-latency dependency for propagation levels one and two. The models trained by our algorithms were trained as declare above but with a difference. For each distance, we trained an individual model with a corresponding constant distance between key and current frames during the training process. Figure 5.1 shows the dependency of level one propagation. As we can see, for all distances our algorithms outperform the truncated version for current and key-frames. Our algorithms are also faster than the baseline.

Figure 5.2 shows the accuracy-latency dependency for two levels of propagation. For current frame masks prediction, not all distances have shown competitive results. Our approach with distance 10 between the key and last current frame has low accuracy rather than the truncated model. The latency at least the same as the truncated has. The latency for distances five and 10 was nearly the same as for truncated one. Moreover, none of the experiments with the intervals could not beat the baseline by the accuracy, but were outperformed by latency.

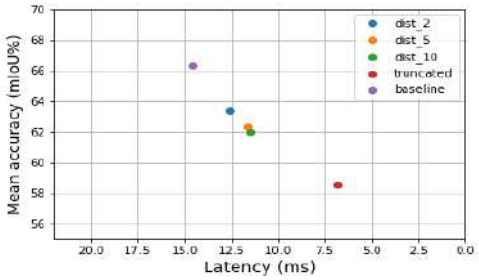

FIGURE 5.4: The comparison of the accuracy and latency between truncated, baseline models and our implementations for constant distances for 2, 5, and 10 frames. The number of propagated levels - 1. The random distance between key and current frame during train process.

Figure 5.3 pictures examples of the predicted semantic segmentation masks for input images by baseline FPN, truncated FPN without two levels and our approach with constant distances two and five. As we can see, those masks predicted by our method with interval two and five generally as well as the baseline except in predicting tiny objects like traffic lights or road signs.

On the other hand, the truncated model had some problems with predicting even bigger objects. It failed to predict a man on a bike (the third row, last image) and the man in the second row has less precise boundaries. It was caused by removing two FPN's levels from the network. Moreover, our models predicted even better than baseline in some cases. For instance, in the first row our model separated a car and bus more precisely, than the baseline did.

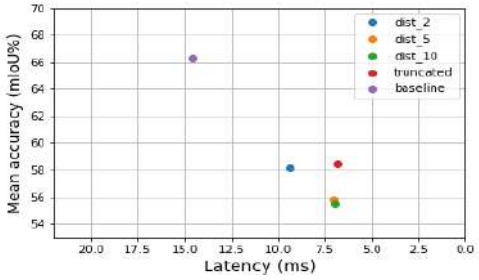

FIGURE 5.5: The comparison of the accuracy and latency between the truncated, baseline models and our implementations for constant distances for 2, 5, and 10 frames. The number of propagated levels - 2. The random distance between key and current frame during train process.

Figure 5.4 and Figure 5.5 illustrate accuracy-latency dependency for one and two levels propagation. The training process for our models was as declared above.

For evaluating accuracy for all three distances, we have trained one model with random distances between key and current frames during the training process. As a result, for tow levels of propagation model the accuracy for current frames underperform the truncated model (see Figure 5.5). For one level propagation model with random distance training, the accuracy for segmenting current frames is lower than models with corresponding propagation level but with constant training distances. Probably it can be caused to the fact that model, with random distance during training, can perform better during flexible key-frame selection. Moreover, such a model can perform worse with constant key-frame distance due to the incorrect frame indication as a key one. It can be checked by applying the efficient methods for selecting key-frames, which is outside the scope of this paper.

According to the results described above, our algorithms have been adapted for the propagating high-level features from key-frames to current. As a result, an FPN transformed by our algorithm with constant training distance has shown competitive results against baseline FPN by accuracy and latency. The low results for predicting masks for current frames for two levels propagation can be caused by caching too many features for propagating (one FPN's level consists of the several convolution maps). Moreover, the models with random training distance have produced accuracy lower then truncated model. The efficient key-frame selector can be used to improve the precision of our architectures with random training distance.

Chapter 6

Conclusions

In this paper, we provided an approach for propagating high-level features from key-frames to currents. We developed the propagation module S^{prop} for sharing high-level features from key-frames to processing current frames. We also developed the loss function L_{total} that helps to teach our architecture network to extract general among frames form one high-level scene features from key-frames and propagate them to current ones. Moreover, L_{total} adapted to count losses for predicting images in a semi-supervised way. We did so, because the extended Cityscapes dataset has the vast number of images without semantic segmentation masks. It has sped up the adapted FPN networks and shown comparable to baseline results.

Due to the results described in chapter 5, the FPN adapted by our approaches have shown competitive results up to baseline FPN. The highest competitive results with respect to a constant distance between key and current frames have shown model training with constant intervals: 2, 5 and 10 images for all propagation levels. They have provided accuracy higher then truncated FPN (more the 58.8% mIoU), even for predicting semantic segmentation mask for current images. The only network with training constant distance 10 has shown lower mIoU then truncated model for two levels of propagation. Our algorithm has also provided the latency decreasing for both methods of choosing the distance between key-current frames and propagation levels. For one propagation level, the latency has reduced at least in 14 % (for our approach with constant distance two between a key and current frames) up to the baseline. On the other hand, for propagation two levels, the latency for our approach has decreased at least for 35 % up to baseline.

The low result was achieved by models trained with random training distance. For all constant distances at the validation set for our architectures with two propagation levels trained with random intervals, the accuracy became lower than the truncated model.

This paper has not discovered the way of the key-frame efficient choosing, efficient process of choosing the number of propagation level, and the direct impact of more complex deep neural networks for the semantic segmentation accuracy. There exist several paths to research for improving the efficiency and latency for our propagating approach. Firstly, there can be developed an efficient method for choosing key-frame independently of the distance between frames. Secondly, PSPnet or DeeplabV3+ or another CNN for semantic segmentation mask prediction with higher mIoU than FPN, for producing artificial semantic segmentation maps for key images during the training process can be used. Thirdly, to develop an approach for finding an optimal number of the cached high-level feature maps.

Bibliography

[1] B. Rhunck B. Horn. "Determining optical flow". In: *Artificial intelligence* 17.1-3 (1981), pp. 185–203.

[2] A. Fern B. Mahasseni S. Todorovic. "Budget-aware deep semantic video segmentation". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2017, pp. 1029–1038.

[3] Christopher M. Bishop. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer-Verlag, 2006.

[4] S. Ioffe J. Shlens C. Szegedy V. Vanhoucke. "Rethinking the inception architecture for computer vision". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 2818–2826.

[5] Florian Schroff Hartwig Adam Wei Hua Alan Yuille Li Fei-Fei Chenxi Liu Liang-Chieh Chen. "Auto-DeepLab:Hierarchical Neural Architecture Search for Semantic Image Segmentation". In: (2019).

[6] Marius Cordts et al. "The cityscapes dataset for semantic urban scene understanding". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 3213–3223.

[7] C. Sminchisescu D. Nilsson. "Semantic Video Segmentation by Gated Recurrent Flow Propagation". In: (2017). URL: https://arxiv.org/pdf/1612.08871.pdf.

[8] Jifeng Dai et al. "Deformable Convolutional Networks". In: *The IEEE International Conference on Computer Vision (ICCV)*. 2017.

[9] J. Deng et al. "ImageNet: A Large-Scale Hierarchical Image Database". In: *CVPR09*. 2009.

[10] J. R. Bergen P. J. Burt J. M. Ogden E. H. Adelson C. H. Anderson. "Pyramid methods in image processing". In: *RCA engineer* 29.6 (1984), pp. 33–41.

[11] Z. Liu. L. Maaten G. Huang. "Densely connected convolutional networks." In: *CVPR*. Vol. 1. 2. 2017, p. 3.

[12] Ian Goodfellow et al. *Deep learning*. Vol. 1. MIT press Cambridge, 2016.

[13] Kaiming He et al. "Deep residual learning for image recognition". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 770–778.

[14] K. Cho Y. Bengio J. Chung C. Gulcehre. "Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling". In: (2014). URL: https://arxiv.org/pdf/1412.3555.pdf.

[15] D.Bahdanau Y. Bengio K. ECho B. "On the properties of neural machine translation: Encoder-decoder approaches". In: (2014).

[16] Diederik P Kingma and Jimmy Ba. "Adam: A method for stochastic optimization". In: *International Conference on Learning Representations* (2015).

[17] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. "Imagenet classification with deep convolutional neural networks". In: *Advances in neural information processing systems*. 2012, pp. 1097–1105.

[18] G. Papandreou F. Schroff H. Adam L. Chen Y. Zhu. "Encoder-Decoder with Atrous Separable Convolution for Semantic Image Segmentation". In: (2018). URL: https://arxiv.org/pdf/1802.02611.pdf.

[19] Tsung-Yi Lin et al. "Feature Pyramid Networks for Object Detection." In: *CVPR*. 2017.

[20] Jonathan Long, Evan Shelhamer, and Trevor Darrell. "Fully convolutional networks for semantic segmentation". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2015, pp. 3431–3440.

[21] M. Sabokrou M. Fathy R. Klette F. Huang M. Fayyaz M. Saffar. "STFCN: spatio-temporal FCN for semantic video segmentation". In: (2016).

[22] Adam Paszke et al. "Automatic differentiation in pytorch". In: (2017).

[23] P. Gehler R. Gadde V. Jampani. "Semantic Video CNNs through Representation Warping". In: (2017). URL: https://arxiv.org/pdf/1708.03088.pdf.

[24] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. "U-net: Convolutional networks for biomedical image segmentation". In: *International Conference on Medical image computing and computer-assisted intervention*. Springer. 2015, pp. 234–241.

[25] J. Schmidhuber S. Hochreiter. "Long short-term memory". In: *Neural computation* 9.8 (1997), pp. 1735–1780.

[26] J. Gonzalez S. Jain X. Wang. "Accel: A Corrective Fusion Network for Efficient Semantic Segmentation on Video". In: (2018). URL: https://arxiv.org/pdf/1807.06667.pdf.

[27] Evan Shelhamer et al. "Clockwork convnets for video semantic segmentation". In: *European Conference on Computer Vision*. Springer. 2016, pp. 852–868.

[28] D. Lin Y. Li J. Shi. "Low-Latency Video Semantic Segmentation". In: *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2018.

[29] Hengshuang Zhao et al. "Pyramid scene parsing network". In: *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*. 2017, pp. 2881–2890.

[30] Bolei Zhou et al. "Scene Parsing through ADE20K Dataset". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2017.

[31] Xizhou Zhu et al. "Deep feature flow for video recognition". In: *CVPR*. 2017.