

# Dynamic Video Segmentation Network

Yu-Syuan Xu, Tsu-Jui Fu\*, Hsuan-Kung Yang\*, *Student Member, IEEE* and Chun-Yi Lee, *Member, IEEE*  
 Elsa Lab, Department of Computer Science, National Tsing Hua University

{yusean0118, rayfu1996zig, hellochick}@gapp.nthu.edu.tw, cylee@cs.nthu.edu.tw

## Abstract

In this paper, we present a detailed design of dynamic video segmentation network (DVSNet) for fast and efficient video semantic segmentation. DVSNet consists of two convolutional neural networks: a segmentation network and a flow network. The former generates highly accurate semantic segmentations, but is deeper and slower. The latter is much faster than the former, but its output requires further processing to generate less accurate semantic segmentations. We explore the use of a decision network to adaptively assign different frame regions to different networks based on a metric called expected confidence score. Frame regions with a higher expected confidence score traverse the flow network. Frame regions with a lower expected confidence score have to pass through the segmentation network. We have extensively performed experiments on various configurations of DVSNet, and investigated a number of variants for the proposed decision network. The experimental results show that our DVSNet is able to achieve up to 70.4% mIoU at 19.8 fps on the Cityscape dataset. A high speed version of DVSNet is able to deliver an fps of 30.4 with 63.2% mIoU on the same dataset. DVSNet is also able to reduce up to 95% of the computational workloads.

## 1. Introduction

Fast and accurate semantic segmentation has been a fundamental challenge in computer vision. The goal is to classify each pixel in an image into one of a given set of categories. In recent years, image semantic segmentation has achieved an unprecedented high accuracy on various datasets [1, 2, 3, 4] via the use of deep convolutional neural networks (DCNNs) [5, 6, 7, 8, 9, 10, 11, 12]. Accurate semantic segmentation enables a number of applications which demand pixel-level precision for their visual perception modules, such as autonomous vehicles [13], surveillance cameras, unmanned aerial vehicles (UAVs), and so on. However, due to their real-time requirements, these applications typically require high frame rates per second (fps), necessitating short inference latency in the perception modules. Unfortunately, contemporary state-of-

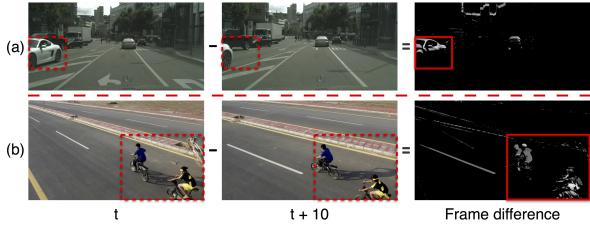


Figure 1: Comparison of frames at timestamps  $t$  and  $t + 10$  in two video sequences

the-art CNN models usually employ deep network architectures to extract high-level features from raw data [14, 15, 16, 17, 18], leading to exceptionally long inference time. The well-known models proposed for image semantic segmentation, including fully convolutional networks (FCN) [19], DeepLab [5, 6, 7], PSPNet [8], ResNet-38 [9], RefineNet [10], dense upsampling convolution (DUC) [11], etc., are not suitable for real-time video semantic segmentation due to their usage of deep network architectures. These models usually incorporate extra layers for boosting their accuracies, such as spatial pyramid pooling (SPP) [6, 7, 8, 20], multi-scale dilated convolution [5, 6, 7, 11, 21], multi-scale input paths [6, 22, 23, 24], multi-scale feature paths [5, 10, 12, 25, 26], global pooling [7], and conditional random field (CRF) [5, 6, 22, 27, 28]. These additional layers consume tremendous amount of computational resources to process every pixel in an image, leading to impractical execution time. In the past decade, video semantic segmentation focusing on reducing the inference time has received little attention [29, 30]. With increasing demand for high accuracy and short inference time, a method for efficiently reusing the extracted features in DCNNs for video semantic segmentation is becoming urgently necessary.

It is unnecessary to reprocess every single pixel of a frame by those deep semantic segmentation models in a video sequence. When comparing the difference between two consecutive frames, it is common that a large portion of them is similar. Fig. 1 illustrates an example of the above observation. The left parts show the video frames at timestamps  $t$  and  $t + 10$ , respectively. The right part shows the difference between these two frames. It can be observed that only a small portion of the frames are apparently different (highlighted by red rectangles), implying that a large portion of the feature maps between these frames is invari-

\*Equal contribution

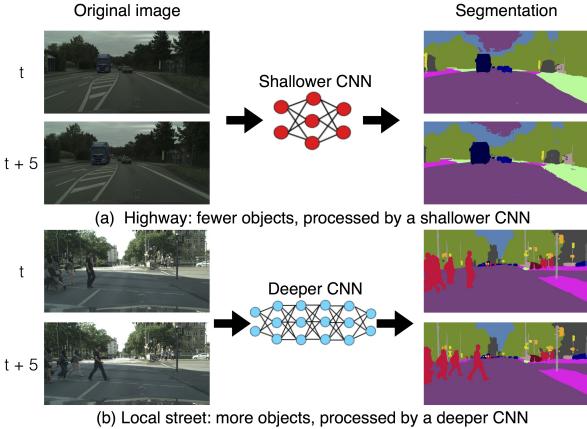


Figure 2: Using different CNNs for different video scenes

ant, or just varies slightly. Therefore, performing complex semantic segmentation on the entire video frame can potentially be a waste of time. By keeping or slightly modifying the feature maps of the portions with minor frame differences while performing semantic segmentation for the rest, we may achieve a better efficiency and shorter latency in video semantic segmentation than per-frame approaches.

Another perspective for accelerating the processing speed of video semantic segmentation is by leveraging the temporal correlations between consecutive frames. Consecutive video frames that do not change rapidly have similar high-level semantic features [31, 32]. On the other hand, frames containing multiple moving objects demonstrate disparate feature maps at different timestamps. Fig. 2 illustrates an example of such scenarios. Fig. 2 (a) shows a semantic segmentation performed on a highway, which contains fewer objects and thus results in less changes in consecutive segmented images. Fig. 2 (b), on the contrary, corresponds to a video sequence taken from a local street, which contains dozens of moving objects. The former suggests reusing the extracted features and updating them with as few computations as possible (e.g., by a shallower CNN), while the latter requires performing highly accurate semantic segmentation on every single frame (e.g., by a deeper CNN). Researchers have attempted several methodologies to reuse the extracted features, such that not all of the frames in a video sequence have to traverse the entire DCNN. The authors in [29] proposed to update high-level features less frequently, taking advantage of temporal correlations between frames to reduce the computation time. Another effective approach is called Deep Feature Flow (DFF) [30], which suggests to propagate the features of a few key frames to other timestamps via the use of optical flow [33]. DFF combines the concepts of segmentation network and FlowNet [34], and is able to reduce the computation time of video semantic segmentation by 74%. Unfortunately, DFF uses a fixed key frame scheduling policy (i.e., it assumes a fixed update period between two consecutive key

frames), which sacrifices its flexibility and customizability.

Based on the above observations, we propose a new network architecture, called dynamic video segmentation network (DVSNet), to adaptively apply two different neural networks to different regions of the frames, exploiting spatial and temporal redundancies in feature maps as much as possible to accelerate the processing speed. One of the networks is called the segmentation network, which generates highly accurate semantic segmentations, but is deeper and slower. The other is called the flow network. The flow network is much shallower and faster than the segmentation network, but its output requires further processing to generate estimated semantic segmentations (which might be less accurate than the ones generated by the segmentation network). The former can be implemented by any of the contemporary state-of-the-art architectures [6, 8], while the latter is developed on top of FlowNet 2.0 [35]. We divide each frame into multiple regions. Regions with minor differences between consecutive frames, where most of the image contents are similar, should traverse the flow network (Fig. 2 (a)). Regions with huge differences between consecutive frames, where the contents change significantly, have to pass through the segmentation network (Fig. 2 (b)). In other words, different regions in a frame may traverse different networks of different lengths when they are presented to DVSNet. We designate the regions processed by the segmentation network and flow network as the key frame regions and spatial warping regions, respectively. In order to accelerate the processing speed, we assume that key frame regions are relatively sparser than the spatial warping regions in a video. DVSNet offers two major advantages. First, efficiency is enhanced because DVSNet adapts its throughput to the differences between consecutive frame regions at runtime. Second, significant computation can be saved by the use of the flow network. This scheme is primarily targeted at video semantic segmentation.

To define a systematic policy for efficiently assign frame regions to the two networks while maintaining flexibility and customizability, we further propose two techniques: (i) adaptive key frame scheduling policy, and (ii) decision network (DN). Adaptive key frame scheduling policy is a technique for determining whether to process an input frame region by the segmentation network or not. Differing from the fixed key frame scheduling policy employed in [30], the proposed adaptive key frame scheduling policy updates the key frames according to a new metric called *expected confidence score*. An *expected confidence score* is evaluated for each frame region. The higher the *expected confidence score* is, the more likely the segmentation generated by the flow network will be similar to that of the segmentation network. The value of the *expected confidence score* reflects the confidence of the flow network to generate similar results as the segmentation network. The larger

the frame region difference is, the more likely the flow network is unable to infer the correct frame region segmentation. Fig. 3 illustrates such a scenario, which is explained in Section 3.1. A frame region is first analyzed for its *expected confidence score*. If its *expected confidence score* is higher than a predefined threshold, it is processed by the flow network. Otherwise, it is allocated to the segmentation network. The decision threshold is adjustable for different scenarios. A higher threshold leads to a higher mean intersection-over-union (mIoU) accuracy of the segmentation results, while a lower thresholds corresponds to a shorter processing latency for most of the frame regions. An adjustable threshold allows DVSNet to be applied to various scenarios. The function of DN is to determine whether an input frame region has to traverse the segmentation network by estimating its *expected confidence score*. In DVSNet, DN is implemented as a CNN, with its network size much smaller than typical network architectures for image recognition. DN can be trained by supervised learning, where the details is covered in Section 3.4. With the use of DN, we are able to obtain fine-grained control of fps, and enhance the efficiency of computation for video semantic segmentation.

To verify the proposed DVSNet, we have extensively performed experiments on several well-know models for the segmentation network, and investigated a number of variants of the proposed DN scheme. The results show that our method is able to achieve up to 70.4% mIoU at 19.8 fps on the Cityscape [2] dataset. A high speed version of DVSNet achieves 30.4 fps with 63.2% mIoU on the same dataset. Our model is able to reduce up to 95% of the computational workloads. The contributions of this work are as follows:

1. A frame division technique to apply different segmentation strategies to different frame regions for maximizing the usage of video redundancy and continuity.
2. A DN for determining whether to assign an input frame region to the segmentation network, and adaptively adjusting the update period of the key frames.
3. An adaptive key frame scheduling policy based on a metric called *expected confidence score*.
4. An adjustable threshold for DN.
5. A comprehensive analysis of the impact of DN’s decision threshold on DVSNet’s accuracy and fps.

The remainder of this paper is organized as follows. Section 2 introduces background material. Section 3 walks through the proposed DVSNet architecture, its implementation details, and the training methodologies. Section 4 presents the experimental results. Section 5 concludes.

## 2. Background

In this section, we introduce background material. We first provide an overview of semantic segmentation and optical flow. Then, we briefly review related work that focuses on video semantic segmentation.

### 2.1. Image Semantic Segmentation

Various techniques have been proposed in the past few years to transfer DCNNs from image classification [14, 15, 16, 17, 36] to image semantic segmentation tasks [7, 8, 9, 10, 19]. Fully convolutional network (FCN) [19] was the pioneer to replace fully-connected layers with convolutional layers. Inspired by FCN, a number of successive methods and modifications were proposed to further improve the accuracy. The authors in [5, 6, 7, 21] investigated the use of dilated (atrous) layers to replace deconvolutional layers [37] in dense prediction tasks, so that the resolution of feature maps can be explicitly controlled. Integrating DCNNs with a fully-connected conditional random field (CRF) have also been explored in [5, 6, 27] to refine the image semantic segmentation results. Spatial pyramid pooling [20] and atrous spatial pyramid pooling (ASPP) [6, 7] are employed by PSPNet [8] and DeepLab [6, 7] to capture multi-scale context information. These methods are effective in increasing the accuracy, however, sometimes at the expense of longer latency and more computational workloads.

### 2.2. Optical Flow

Optical flow estimation has been the subject of a number of research works [34, 33, 38, 39, 40, 41, 42]. Unfortunately, most of the previous methodologies are mainly developed for running on CPUs, failing to incorporate execution efficiency offered by GPUs. For deep learning approaches running on GPUs, FlowNet [34] is the first model to apply DCNNs to optical flow estimation. It then later evolves into two recent architectures. One is called spatial pyramid network (SpyNet) [43], which uses the coarse-to-fine spatial pyramid structure of [44] to learn residual flow at each pyramid level. The other is FlowNet 2.0 [35], which introduces a new learning schedule, a stacked architecture, and a sub-network specialized on small motions to enhance flow estimation. In this paper, we incorporate the building blocks from FlowNet 2.0 into our DVSNet for accelerating video semantic segmentation.

### 2.3. Video Semantic Segmentation

Video semantic segmentation has gained researchers’ attention in recent years. Many research works are focused on efficient network architecture for speeding up video semantic segmentation [29, 30]. Clockwork employs different update periods for different layers of the feature maps in the network, and reuses feature maps of past frames in certain network layers to reduce computation [29]. Deep feature flow (DFF) exploits an optical flow network to generate flow fields and propagates feature maps from key frames to nearby frames [30]. It is reported that Clockwork runs 1.3× faster than per-frame approach [29], however, its mIoU drops from 65.9% to 64.4% on the Cityscapes dataset [2].

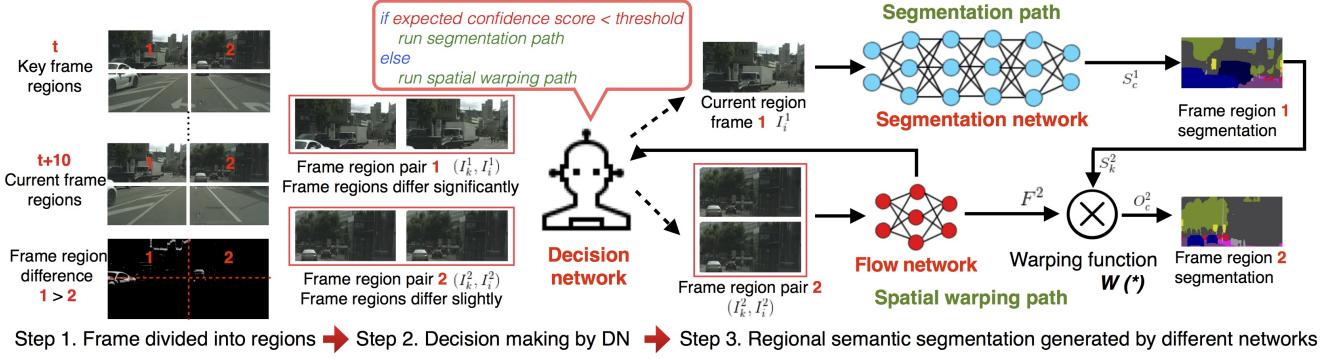


Figure 3: DVSNet framework

In contrast, DFF runs three times faster than per-frame approach, and its mIoU only drops slightly from 71.1% to 70.0% on the same datasets [30]. DFF demonstrates much better performance than Clockwork in both accuracy and efficiency. However, a major drawback is that it employs a fixed key frame scheduling policy. Inspired by DFF, the proposed DVSNet embraces an adaptive key frame scheduling policy, offering better performance than DFF in terms of both accuracy and efficiency.

### 3. DVSNet

In this section, we present the architecture and implementation details of DVSNet. We first outline the framework of DVSNet. Next, we introduce its methodologies including adaptive key frame scheduling, frame region based execution. Finally, we explain the model of DN and its training methodology in detail.

#### 3.1. Dynamic Video Segmentation Network

The framework of DVSNet is illustrated in Fig. 3. The DVSNet framework consists of three major steps. The first step in the DVSNet framework is dividing the input frames into frame regions. In Fig. 3, we assume that  $I_k$  represents the key frame,  $I_i$  represents the current frame, and the number of the frame regions equals four. We further assume that the frame regions at timestamp  $t$  correspond to the key frame regions, and those at timestamp  $t + 10$  correspond to the current frame regions. The differences between the key frame regions and the corresponding current frame regions are shown at the bottom left. In this example, region 1 shows significantly more differences in pixels between timestamps  $t$  and  $t + 10$ , while the other regions only change slightly. In step 2, DN analyzes the frame region pairs between  $I_k$  and  $I_i$ , and evaluates the *expected confidence scores* for the four regions separately. DN compares the *expected confidence score* of each region against a pre-determined threshold. If the *expected confidence score* of a region is lower than the threshold, the corresponding region is sent to a segmentation path (i.e., the segmentation network). Otherwise, it is forwarded to a spatial warping path,

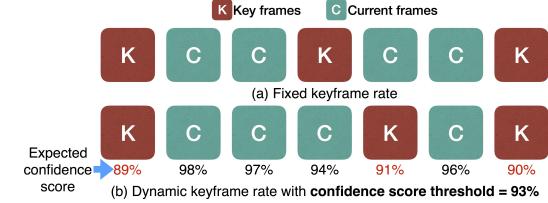


Figure 4: Different key frame scheduling policies

which includes the flow network. The function of DN is to evaluate if the spatial warping path is likely to generate similar segmentation results ( $O_c$ ) as the segmentation path ( $S_c$ ). The higher the *expected confidence score* is, the more likely the spatial warping path is able to achieve it. We explain the training methodology of DN in Section 3.4. Based on the decisions of DN, in step 3 frame regions are forwarded to different paths to generate their regional semantic segmentations. For the spatial warping path, a special warping function  $W(*)$  [30] is employed to process the the output of the flow network  $F$  with the segmentation  $S_k$  from the same region of the key frame to generate a new segmentation  $O_c$  for that region. Please note that the flow network can not generate a regional semantic segmentation by itself. It simply predicts the displacement of objects by optical flow, and needs to rely on the warping function  $W(*)$  and the information contained in the key frames. We recommend interested readers to refer to [30] for more details of  $W(*)$ .

#### 3.2. Adaptive Key Frame Scheduling

Fig. 4 illustrates the key frame scheduling policies used by DFF [30] and DVSNet. We assume that the sequences of the frame regions in Fig. 4 correspond to the same frame region  $r$ . Similar to DFF, DVSNet updates the key frames after a certain period of time. DFF adopts a fixed update period, as shown in Fig. 4 (a), which is predetermined and does not take quality and efficiency into consideration. For example, it is more efficient to process a frame sequence of similar contents with a longer update period, as the spatial warping path itself is sufficient to produce satisfactory outcomes (i.e., regional segmentations). On the other hand, when the scene changes dramatically, using the segmenta-

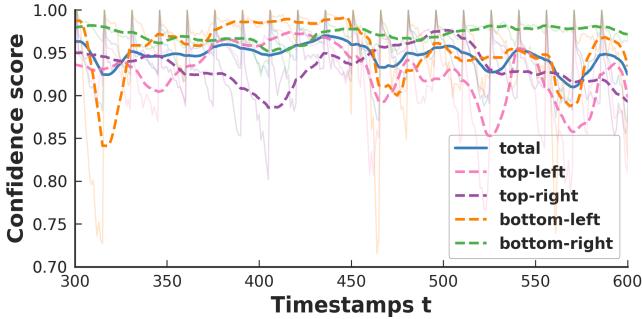


Figure 5: Confidence score vs.  $t$  for the frame regions and the entire frame

tion network is more reasonable. This is because the flow network is unable to predict the displacement of unseen objects not existing in the corresponding key frame region.

We propose an adaptive key frame scheduling policy by using DN and *expected confidence score*. The adaptive key frame scheduling policy is illustrated in Fig. 4 (b), in which the update period is not fixed and is determined according to the *expected confidence score* of that region. DN determines when to update the key frame region  $r$  by evaluating if the output of the flow network  $F^r$  is able to generate a satisfactory regional segmentation  $O^r$ . If  $O^r$  is expected to be close to that of the segmentation network  $S^r$ ,  $F^r$  is forwarded to the spatial warping function  $W(*)$  to generate  $O^r$ . Otherwise, the current frame region  $I_c^r$  is sent to the longer segmentation network, and the key frame is updated ( $I_k^r <= I_c^r$ ). Note that DN neither compares  $O^r$  and  $S^r$ , nor requires them in its evaluation. It is a regression model trained to "predict" the outcome of  $O^r$  based on  $F^r$ , as illustrated in Fig 6, and is explained in Section 3.4. We define a metric, called *confidence score*, to represent the ground truth difference between  $O^r$  and  $S^r$ . Please note that *expected confidence score* and *confidence score* are different. The former is a value evaluated by DN, while the latter is the ground truth difference in pixels between  $O^r$  and  $S^r$ . The latter is only used in the training phase for training DN, and is not accessible by DN during the execution phase. The mathematical form of *confidence score* is defined as:

$$\text{confidence score} = \frac{\sum_{p \in P} C(O^r(p), S^r(p))}{P} \quad (1)$$

where  $P$  is the total number of pixels in  $r$ ,  $p$  the index of a pixel,  $O^r(p)$  the class label of pixel  $p$  predicted by the spatial warping path,  $S^r(p)$  the class label of pixel  $p$  predicted by the segmentation path,  $C(u, v)$  a function which outputs 1 only when  $u$  equals  $v$ , otherwise 0.

Given a target threshold  $t$ , DN compares its *expected confidence score* against  $t$ . If it is higher than  $t$ ,  $F^r$  is considered satisfactory. Otherwise,  $I^r$  is forwarded to the segmentation path in Fig 3. An advantage of the proposed adaptive policy is that the target threshold  $t$  is adjustable. A lower  $t$  leads to lower accuracy and higher fps, as more in-

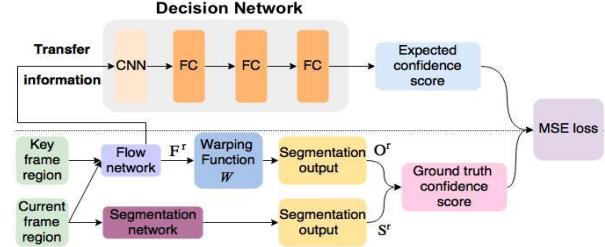


Figure 6: The network model of DN and its training methodology

put frame regions traverse the shorter spatial warping path. On the other hand, a higher  $t$  results in higher accuracy, trading off speed for quality. According to different requirements in different scenes and scenarios, DVSNet can be customized to determine the best  $t$  value.

### 3.3. Frame Region Based Execution

We provide an analytical example to justify the proposed frame region based execution scheme. Fig. 5 plots curves representing the values of *confidence score* versus time for different frame regions as well as the entire frame for a video sequence extracted from the Cityscape dataset [2]. We again assume that each frame is divided into four frame regions. We plot the curves from timestamp 300 to 600 and use fixed a key frame scheduling policy which updates the key frame every 15 frames. The curves are smoothed by averaging the data points over 15 timestamps. The smoothed curves are highlighted in solid colors, while the raw data points are plotted in light colors. It can be seen that the *confidence score* of the entire frame does not fluctuate obviously over time. However, for most of the time, the *confidence scores* of different frame regions show significant variations. Some frame regions exhibit high *confidence scores* for a long period of time, indicating that some portions of the frame change slowly during the period. For those scenarios, it is not necessary to feed the entire frame to the segmentation network. This example validates our claim of using frame region based execution scheme. In our experimental results, we present a comprehensive analysis for the number of frame regions versus performance. We also inspect the impact of overlapped pixels between frame regions on DVSNet's accuracy.

### 3.4. DN and Its Training Methodology

Fig. 6 illustrates the network model of DN as well as its training methodology. DN is a lightweight CNN consists of only a single convolutional layer and three fully-connected layers. DN takes as input the feature maps from one of the intermediate layers of the flow network, as illustrated in Fig. 7. DN is trained to perform regression. In the training phase, the goal of DN is to learn to predict an *expected confidence score* for a frame region as close to the ground truth *confidence score* (derived from Eq. (1)) of that region as possible. The predicted *expected confidence score*

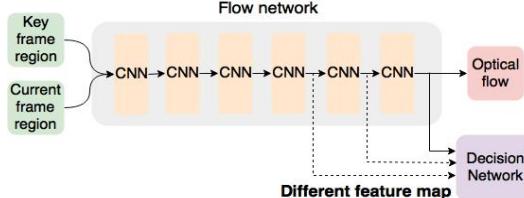


Figure 7: Different feature maps for training DN

is compared with the ground truth *confidence score* to calculate a mean squared error (MSE) loss. The MSE loss is then used to update the parameters in the model of DN by Adam optimizer [45]. In the execution (testing) phase, the ground truth *confidence score* is not accessible to both DN and the flow network. The feature maps fed into DN is allowed to come from any of the layers of the flow networks, as plotted in Fig. 7. These feature maps represent the spatial transfer information between a key frame region and its corresponding current frame region [30, 46]. We provide an analysis in Section 4.4 for different DN configurations.

## 4. Experiments

In this section, we present experimental results and discuss their implications. We start by a brief introduction to our experimental setup in Section 4.1. Then, we validate our DVSNet for a variety of configurations in Section 4.2. We demonstrate the effectiveness of the proposed adaptive key frame scheduling policy in Section 4.3. We compare different DN configurations in Section 4.4. We analyze different frame division schemes in Section 4.5. Finally, we evaluate the impact of overlapping frame regions in Section 4.6.

### 4.1. Experimental Setup

We perform experiments on the famous Cityscapes dataset [2]. The training, validation, and testing sets contain 2,975, 500, and 1,525 frames, respectively. The frames of training and validation sets are annotated with pixel-level ground-truth labels for semantic segmentation. The annotated frame is provided on the 20<sup>th</sup> frame of a 30-frame video snippet. We evaluate our approaches on each Cityscapes validation snippet from the 1<sup>st</sup> frame to the 20<sup>th</sup> frame. We set the 1<sup>st</sup> frame as our initial key frame and measure mIoU on the annotated 20<sup>th</sup> frames.

In our experiments, we pre-trained three semantic segmentation models, DeepLab-Fast, PSPNet, and DeepLab-v2, as our baseline models for the segmentation network in DVSNet. DeepLab-Fast is a modified version of DeepLab-v2 [6], while PSPNet and DeepLab-v2 are reproduced from PSPNet [8] and DeepLab-v2 [6], respectively. For the reproduced baseline models, we removed several extra features including CRF, multi-scale inferencing, and sliding window segmentation from the original versions to enhance their execution speed. The values of mIoU and fps of the baseline segmentation models are measured in a per-frame

DVSNet	Methods	mIoU (%)	fps
Baseline segmentation models			
<i>DeepLab-Fast</i>	per-frame	73.5	5.6
<i>PSPNet</i>	per-frame	77.0	1.7
<i>DeepLab-v2</i>	per-frame	74.8	1.8
Balanced mode			
( <i>DeepLab-Fast</i> , <i>FlowNet2-s</i> )	$t = 92\%$	70.4	19.8
( <i>PSPNet</i> , <i>FlowNet2-s</i> )	$t = 83\%$	70.2	11.5
( <i>DeepLab-v2</i> , <i>FlowNet2-s</i> )	$t = 81\%$	70.3	8.3
High-speed mode			
( <i>DeepLab-Fast</i> , <i>FlowNet2-s</i> )	$t = 86\%$	63.2	30.4
( <i>PSPNet</i> , <i>FlowNet2-s</i> )	$t = 75\%$	62.6	30.3

Table 1: Comparison of mIoU and fps for various models, where  $t$  represents the target threshold of DN.

fashion, without any frame division or assistance of the flow network. The results of the baseline segmentation models on the Cityscape dataset are summarized in Table 1. Among the three baseline segmentation models, DeepLab-Fast is three times faster than the other two models, while PSPNet has the highest 77.0% mIoU accuracy. We further pre-trained FlowNet2-S and FlowNet2-s to serve as our baseline models for the flow network in DVSNet. These two models are reproduced from [35]. In this paper, we represent the DVSNet configuration by a tuple: (*segmentation network*, *flow network*,  $t$ ), where  $t$  is the target threshold for that DVSNet configuration. By default, we divide a frame to four regions for the former two models, and two regions for the latter model. The depth of the overlapped regions between adjacent frame regions is by default set to 64 pixels. A detailed analysis for justifying this value is provided in Section 4.6. We perform all of our experiments on a server with two Intel Xeon E5-2620 CPUs and an NVIDIA GTX 1080 Ti GPU.

### 4.2. Validation of DVSNet

Table 1 compares the speed (fps) and accuracy (mIoU) of (*DeepLab-Fast*, *FlowNet2-s*), (*PSPNet*, *FlowNet2-s*), and (*DeepLab-v2*, *FlowNet2-s*) for two different modes: a balanced mode and a high-speed mode. The balanced mode requires that the accuracy of a network has to be above 70% mIoU, while the high-speed mode requires that the frame rate has to be higher than 30 fps. We show the corresponding values of the target threshold  $t$  in the 2<sup>nd</sup> column, and the values of mIoU and fps in the 3<sup>rd</sup> and 4<sup>th</sup> columns, respectively. It is observed that the DVSNet framework is able to significantly improve the performance of the three baseline models. For the balanced mode, (*DeepLab-Fast*, *FlowNet2-s*, 92), (*PSPNet*, *FlowNet2-s*, 83), and (*DeepLab-v2*, *FlowNet2-s*, 81) are 3.5×, 6.8×, and 4.6× faster than their baseline counterparts, with a slight drop of 3%, 6%, and 4% in mIoU, respectively. The high-speed mode enables the two models to achieve real-time speed. The frame rates of (*DeepLab-Fast*, *FlowNet2-s*, 86) and (*PSP-*

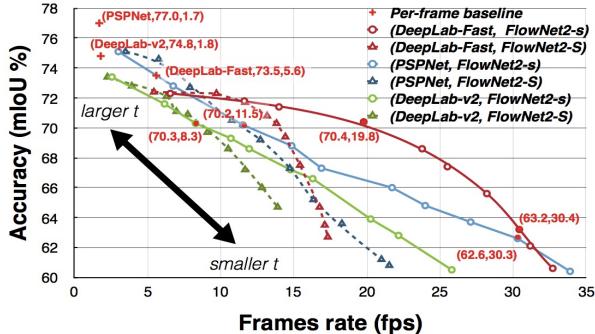


Figure 8: Accuracy (mIoU) and frame rate (fps) of various DVS-Net configurations under different threshold  $t$ .

*Net, FlowNet2-s, 75)* are  $5.4\times$  and  $17.8\times$  faster than their baseline counterparts, respectively. The mIoU of them declines to 63.2% and 62.6%, respectively. From Table 1, we conclude that decreasing  $t$  leads to a drop in mIoU of the models, but increases fps significantly.

Fig. 8 shows accuracy (mIoU) versus frame rate (fps) for various DVSNet configurations. We plot six curves on Fig. 8, corresponding to six possible combinations of the three baseline segmentation network models and the two baseline flow network models. The three solid curves represent DVSNet configurations built on top of FlowNet2-s, while the remaining three dashed curves stand for DVSNet configurations using FlowNet2-S. The data points on each curve correspond to different values of target threshold  $t$  under the same DVSNet configuration. It can be observed that as  $t$  increases, the data points of all curves move toward the upper-left corner, leading to increased mIoU accuracies but decreased fps for all DVSNet configurations. On the contrary, when  $t$  decreases, the data points of all curves move toward the opposite bottom-right corner, indicating that more frame regions pass through the shorter spatial warping path. It can be seen that the dashed lines drop faster than the solid lines, because FlowNet2-S is deeper and thus slower than FlowNet2-s. By adjusting the value of  $t$  and selecting the baseline models, DVSNet can be configured and customized to meet a wide range of accuracy and frame rate requirements.

#### 4.3. Validation of DVSNet’s Adaptive Key Frame Scheduling Policy

In this section, we validate the effectiveness of DVSNet’s adaptive key frame scheduling policy. Fig. 9 plots a comparison of performance between the fixed key frame scheduling policy and the adaptive key frame scheduling policy. DVSNet which adopts adaptive scheduling with *expected confidence score* and DFF [30] which adopts fixed scheduling correspond to the red and blue curves, respectively. The evaluation method of the fixed key frame scheduling policy employed by DFF tends to overestimate mIoU. For each frame  $i$  with ground truth annotation, DFF aver-

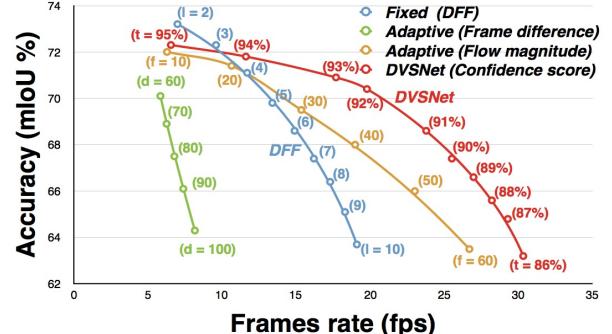


Figure 9: Accuracy (mIoU) versus frame rate (fps) under different key frame scheduling policies.  $t$  is the target confidence score threshold.  $l$  is the key frame update period in DFF [30].  $d$  is the frame difference threshold.  $f$  is the flow magnitude threshold.

ages mIoU from  $m$  image pairs  $(k, i)$ , where key frames  $k = i - (m + 1), \dots, i$ . The method seems counter-intuitive and fails to reflect the impact of scheduling policies on mIoU. In our fixed key frame scheduling policy experiment, we start from the key frames  $k = i - (l + 1)$ , where  $l$  is key frame update period, and measure mIoU at each frame  $i$ .

We include additional curves in Fig. 9 to compare the impact of another two different decision metrics for the adaptive key frame scheduling policy: frame difference (green curve) and flow magnitude (orange curve). The frame difference  $d$  and the flow magnitude  $f$  are given by:

$$d = \frac{\sum_{p \in P} (G(I_k)_p - G(I_i)_p)}{P} \quad (2)$$

$$f = \frac{\sum_{p \in P} (\sqrt{u_p^2 + v_p^2})}{P} \quad (3)$$

where  $P$  is the total number of pixels in a frame or frame region,  $p$  represents the index of a pixel,  $G(*)$  is a grayscale operator which converts an RGB image to a grayscale one, and  $u$  and  $v$  represent the horizontal and vertical movements, respectively. For a fair comparison, the networks are all configured to (*DeepLab-Fast, FlowNet2-s*).

Fig. 9 reveals that using frame difference as the decision metric for the adaptive key frame scheduling policy is inadequate. For all values of  $d$  considered in our experiments (ranging from 60 to 100), the mIoU accuracies and the frame rates of the datapoints on the green curve are much lower than those of the other three curves. On the other hand, it is observed that using the adaptive key scheduling policy with either *expected confidence score* or flow magnitude as the decision metrics deliver higher mIoU accuracies than the fixed key frame scheduling policy employed by DFF, even at high fps. The curves indicate that DVSNet employing the adaptive scheduling policy with *expected confidence score* as the decision metric results in the best performance. This experiment validates the effectiveness of DVSNet’s adaptive key frame scheduling policy.

Inputs fed to DN	Error rate(%)
key-frame + flow	2.20
Feature maps after conv4	1.94
Feature maps after conv5	1.85
Feature maps after conv6	<b>1.74</b>

Table 2: Impact of input feature maps on DN’s performance

Dividing Methods	Spatial Warping / Segmenation
Original	$4,535/5,465 = 0.83$
Half	$9,798/10,202 = 1.04$
$2 \times 2$	$34,013/5,987 = \mathbf{5.68}$
$3 \times 3$	$69,157/20,843 = 3.32$
$4 \times 4$	$122,958/37,042 = 3.32$

Table 3: Impact of frame division schemes on performance

#### 4.4. Comparison of DN Configurations

We perform experiments for a number of DN configurations illustrate in Fig. 7, and summarize the results in Table 2. We feed DN with feature maps from different layers of the flow network, including the feature maps after the 4<sup>th</sup>, 5<sup>th</sup>, and 6<sup>th</sup> convolutional layers. We additional include an entry in Table 2 to represent the case where DN is fed with the key frame and the output of the flow network. We measure the performance of these variants by comparing the error rate of the evaluated *expected confidence score* against the ground truth *confidence score*. It is observed that the feature maps after the 6<sup>th</sup> convolutional layers lead to the lowest error rate.

#### 4.5. Impact of Frame Division Schemes

Table 3 shows the impact of different frame division schemes on performance. Table 3 compares five different schemes corresponding to five different ways to divide a video frame. The ratio of the number of frame regions sent to the spatial warping path to the number of those sent to the segmentation path is represented as *Spatial Warping/Segmentation*. A higher ratio indicates that more frame regions are forwarded to the spatial warping path. We observe that the  $2 \times 2$  division scheme offers the best utilization of the spatial warping path. The results also show that too few or too many frame regions do not exploit the spatial warping path well. A frame without any frame regions prohibits DVSNet from exploiting non-homogeneity in frame differences between consecutive frames. Too many frame regions lead to large variations in *expected confidence score*, causing many of them to be forwarded to the segmentation network frequently.

#### 4.6. Impact of Overlapped Regions on Accuracy

Fig 10 shows the accuracy of DVSNet as a function of the depth of overlapped frame regions. We assume that each video frame is divided to four frame regions. The configuration of DVSNet is set to be (*DeepLab-Fast*, *FlowNet2-s*, 92). The x-axis represents the overlapped depth in pixels

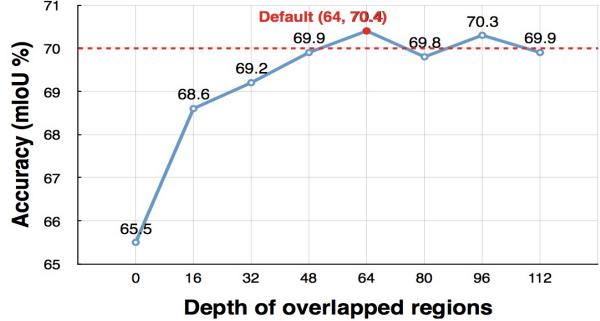


Figure 10: Impact of overlapping frame regions on accuracy

between adjacent frame regions. The y-axis represents the mIoU accuracy. Without overlapped regions, DVSNet is only able to achieve an accuracy up to 65.5% mIoU. However, when the depth of the overlapped region increases, the mIoU accuracy also increases. The curve saturates at 70% mIoU, corresponding to an overlapped depth of 64 pixels. However, more overlap pixels lead to larger frame regions, resulting in increased computation time. According to our experiments, an empirical value of 64 pixels seem to be optimal for DVSNet. We plot a red dotted line corresponding to the mIoU accuracy of the entire frame without any frame division in Fig 10 for the purpose of comparison.

## 5. Conclusion

We presented a DVSNet framework to strike a balance between quality and efficiency for video semantic segmentation. The DVSNet framework consists of two major parts: a segmentation path and a spatial warping path. The former is deeper and slower but highly accurate, while the latter is faster but less accurate. We proposed to divide video frames into frame regions, and perform semantic segmentation for different frame regions by different DVSNet paths. We explored the use of DN to determine which frame regions should be forwarded to which DVSNet paths based on a metric called *expected confidence score*. We further proposed an adaptive key frame scheduling policy to adaptively adjust the update period of key frames at runtime. Experimental results show that DVSNet is able to achieve up to 70.4% mIoU at 19.8 fps on the Cityscape dataset. We have performed extensive experiments for various configurations of DVSNet, and showed that DVSNet outperforms contemporary state-of-the-art semantic segmentation models in terms of efficiency and flexibility.

## 6. Acknowledgments

The authors thank MediaTek Inc. for their support in researching funding, and NVIDIA Corporation for the donation of the Titan X Pascal GPU used for this research.

## References

- [1] M. Everingham *et al.*, “The PASCAL visual object classes challenge: A retrospective,” *Int. J. Computer Vision*, vol. 111, no. 1, pp. 98-136, Jan. 2015. [1](#)
- [2] M. Cordts *et al.*, “The cityscapes dataset for semantic urban scene understanding,” in *Proc. IEEE Conf. Computer Vision and Pattern Recognition (CVPR)*, pp. 3213-3223, Jun. 2016. [1, 3, 5, 6](#)
- [3] B. Zhou *et al.*, “Scene parsing through ADE20K dataset,” in *Proc. IEEE Conf. Computer Vision and Pattern Recognition (CVPR)*, pp. 5122-5130, Jul. 2017. [1](#)
- [4] L. Tsung-Yi and other, “Microsoft COCO: Common objects in context,” in *Proc. European Conf. Computer Vision (ECCV)*, pp. 740-755, Sep. 2014. [1](#)
- [5] 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 *Proc. Int. Conf. Learning Representations (ICLR)*, May 2015. [1, 3](#)
- [6] 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 Trans. Pattern Analysis and Machine Intelligence (TPAMI)*, Apr. 2017. [1, 2, 3, 6](#)
- [7] L.-C. Chen, G. Papandreou, S. F, and H. Adam, “Rethinking atrous convolution for semantic image segmentation,” *arXiv:1706.0558*, Aug. 2017. [1, 3](#)
- [8] H. Zhao, J. Shi, X. Qi, X. Wang, and J. Jia, “Pyramid scene parsing network,” in *Proc. IEEE Conf. Computer Vision and Pattern Recognition (CVPR)*, pp. 6230-6239, Jul. 2017. [1, 2, 3, 6](#)
- [9] Z. Wu, C. Shen, and A. van den Hengel, “Wider or deeper: Revisiting the resnet model for visual recognition,” *arXiv:11611.10080.*, Nov. 2016. [1, 3](#)
- [10] G. Lin, A. Milan, C. Shen, and I. Reid, “Refinenet: Multi-path refinement networks for high-resolution semantic segmentation,” in *Proc. IEEE Conf. Computer Vision and Pattern Recognition (CVPR)*, pp. 5168-5177, Jul. 2017. [1, 3](#)
- [11] P. Wang *et al.*, “Understanding convolution for semantic segmentation,” *1702.08502*, Feb. 2017. [1](#)
- [12] G. GhiasiEmail and C. C. Fowlkes, “Laplacian pyramid reconstruction and refinement for semantic segmentation,” in *Proc. European Conf. Computer Vision (ECCV)*, pp. 519-534, Oct. 2016. [1](#)
- [13] D. Alexey, R. German, C. Felipe, L. Antonio, and K. Vladlen, “CARLA: An open urban driving simulator,” in *Proc. Conf. on Robot Learning (CoRL)*, pp. 445-461, Nov. 2017. [1](#)
- [14] A. Krizhevsky, I. Sutskever, and G. E. Hinton, “ImageNet classification with deep convolutional neural networks,” in *Proc. Neural Information Processing Systems (NIPS)*, pp. 1097-1105, Dec. 2012. [1, 3](#)
- [15] K. Simonyan and A. Zisserman, “Very deep convolutional networks for large-scale image recognition,” in *Proc. Int. Conf. Learning Representations (ICLR)*, May 2015. [1, 3](#)
- [16] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” in *Proc. IEEE Conf. Computer Vision and Pattern Recognition (CVPR)*, pp. 770-778, Jun. 2016. [1, 3](#)
- [17] C. Szegedy *et al.*, “Going deeper with convolutions,” in *Proc. IEEE Conf. Computer Vision and Pattern Recognition (CVPR)*, pp. 1-9, Jun. 2015. [1, 3](#)
- [18] C. Szegedy, S. Ioffe, V. Vanhoucke, and A. A. Alemi, “Inception-v4, inception-resnet and the impact of residual connections on learning,” in *Proc. Association for the Advancement of Artificial Intelligence (AAAI)*, pp. 4278-4284, Feb. 2017. [1](#)
- [19] J. Long, E. Shelhamer, and T. Darrell, “Fully convolutional networks for semantic segmentation,” in *Proc. IEEE Conf. Computer Vision and Pattern Recognition (CVPR)*, pp. 3431-3440, Jun. 2015. [1, 3](#)
- [20] K. He, X. Zhang, S. Ren, and J. Sun, “Spatial pyramid pooling in deep convolutional networks for visual recognition,” *IEEE Trans. Pattern Analysis and Machine Intelligence (TPAMI)*, vol. 37, no. 9, pp. 1904-1916, Sep. 2015. [1, 3](#)
- [21] F. Yu and V. Koltun, “Multi-scale context aggregation by dilated convolutions,” in *Proc. Int. Conf. Learning Representations (ICLR)*, May 2016. [1, 3](#)
- [22] J. Long, E. Shelhamer, and T. Darrell, “Efficient piecewise training of deep structured models for semantic segmentation,” in *Proc. IEEE Conf. Computer Vision and Pattern Recognition (CVPR)*, pp. 3194-3203, Jun. 2016. [1](#)
- [23] H. Zhao, X. Qi, X. Shen, J. Shi, and J. Jia, “ICNet for real-time semantic segmentation on high-resolution images,” *arXiv:1704.08545.*, Apr. 2017. [1](#)
- [24] L.-C. Chen, Y. Yang, J. Wang, W. Xu, and A. L. Yuille, “Attention to scale: Scale-aware semantic image segmentation,” in *Proc. IEEE Conf. Computer Vision and Pattern Recognition (CVPR)*, pp. 3640-3649, Jun. 2016. [1](#)
- [25] T. Pohlen, A. Hermans, M. Mathias, and B. Leibe, “Full-resolution residual networks for semantic segmentation in street scenes,” in *Proc. IEEE Conf. Computer Vision and Pattern Recognition (CVPR)*, pp. 3309-3318, Jul. 2017. [1](#)
- [26] S. Zagoruyko *et al.*, “A multipath network for object detection,” *arXiv:1604.02135*, Aug. 2016. [1](#)
- [27] S. Zheng *et al.*, “Conditional random fields as recurrent neural networks,” in *Proc. IEEE Int. Conf. Computer Vision (ICCV)*, pp. 1529-1537, Dec. 2015. [1, 3](#)
- [28] P. Krähenbühl and V. Koltun, “Efficient inference in fully connected crfs with gaussian edge potentials,” in *Proc. Neural Information Processing Systems (NIPS)*, pp. 109-117, Dec. 2011. [1](#)
- [29] E. Shelhamer, K. Rakelly, J. Hoffman, and T. Darrell, “Clockwork convnets for video semantic segmentation,” in *Proc. European Conf. Computer Vision (ECCV) Wksp*, pp. 852-868, Oct. 2016. [1, 2, 3](#)
- [30] X. Zhu, Y. Xiong, J. Dai, L. Yuan, and Y. Wei, “Deep feature flow for video recognition,” in *Proc. IEEE Conf. Computer Vision and Pattern Recognition (CVPR)*, pp. 4141-4150, Jul. 2017. [1, 2, 3, 4, 6, 7](#)

- [31] L. Wiskott and T. J. Sejnowski, “Slow feature analysis: Unsupervised learning of invariances,” *Neural computation*, vol. 14, no. 4, pp. 715-770, Apr. 2002. [2](#)
- [32] M. D. Zeiler and R. Fergus, “Slow and steady feature analysis: Higher order temporal coherence in video,” in *Proc. IEEE Conf. Computer Vision and Pattern Recognition (CVPR)*, pp. 3852-3861, Jun. [2](#)
- [33] B. K. P. Horn and B. G. Schunck, “Determining optical flow,” *J. Artificial intelligence*, vol. 17, no. 1-3, pp. 185-203, Aug. 1981. [2, 3](#)
- [34] A. Dosovitskiy *et al.*, “FlowNet: Learning optical flow with convolutional networks,” in *Proc. IEEE Int. Conf. Computer Vision (ICCV)*, pp. 2758-2766, Dec. 2015. [2, 3](#)
- [35] E. Ilg *et al.*, “FlowNet 2.0: Evolution of optical flow estimation with deep networks,” in *Proc. IEEE Conf. Computer Vision and Pattern Recognition (CVPR)*, pp. 1647-1655, Jul. 2017. [2, 3, 6](#)
- [36] S. Ioffe and C. Szegedy, “Batch Normalization: accelerating deep network training by reducing internal covariate shift,” in *Proc. Machine Learning Research (PMLR)*, vol. 37, pp. 448-456, Jul. 2015. [3](#)
- [37] M. D. Zeiler, D. Krishnan, G. W. Taylor, and R. Fergus, “Deconvolutional networks,” in *Proc. IEEE Conf. Computer Vision and Pattern Recognition (CVPR)*, pp. 2528-2535, Jun. 2010. [3](#)
- [38] J. Weickert, A. Bruhn, T. Brox, and N. Papenberg, “A survey on variational optic flow methods for small displacements,” *Mathematical Models for Registration and Applications to Medical Imaging*, pp. 103-136, Oct. 2006. [3](#)
- [39] T. Brox and J. Malik, “Large displacement optical flow: Descriptor matching in variational motion estimation,” *IEEE Trans. Pattern Analysis and Machine Intelligence (TPAMI)*, vol. 33, no. 3, pp. 500-513, May 2011. [3](#)
- [40] P. Weinzaepfel, J. Revaud, Z. Harchaoui, and C. Schmid., “DeepFlow: Large displacement optical flow with deep matching,” in *Proc. IEEE Conf. Computer Vision and Pattern Recognition (CVPR)*, pp. 1385-1392, Dec. 2013. [3](#)
- [41] C. Bailer, B. Taetz, and D. Stricker, “Flow fields: Dense correspondence fields for highly accurate large displacement optical flow estimation,” in *Proc. IEEE Int. Conf. Computer Vision (ICCV)*, pp. 4015-4023, Dec. 2015. [3](#)
- [42] J. Wulff and M. J. Black, “Efficient sparse-to-dense optical flow estimation using a learned basis and layers,” in *Proc. IEEE Conf. Computer Vision and Pattern Recognition (CVPR)*, pp. 120-130, Jun. 2015. [3](#)
- [43] A. Ranjan and M. J. Black, “Optical flow estimation using a spatial pyramid network,” *arXiv:1611.00850.*, Nov. 2016. [3](#)
- [44] E. L. Denton, S. Chintala, R. Fergus, *et al.*, “Deep generative image models using a laplacian pyramid of adversarial networks,” in *Proc. Neural Information Processing Systems (NIPS)*, pp. 1486-1494, Dec. 2015. [3](#)
- [45] D. P. Kingma and J. Ba, “Adam: A method for stochastic optimization,” in *Proc. Int. Conf. Learning Representations (ICLR)*, May 2015. [6](#)
- [46] D. Jayaraman and K. Grauman, “Visualizing and understanding convolutional networks,” in *Proc. European Conf. Computer Vision (ECCV)*, pp. 818-833, Sep. 2014. [6](#)