Semantic Video CNNs through Representation Warping

Raghudeep Gadde[1,3], Varun Jampani[1,4] and Peter V. Gehler[1,2,3]
[1]MPI for Intelligent Systems, [2]University of Würzburg
[3]Bernstein Center for Computational Neuroscience, [4]NVIDIA
{raghudeep.gadde,varun.jampani,peter.gehler}@tuebingen.mpg.de

Abstract

In this work, we propose a technique to convert CNN models for semantic segmentation of static images into CNNs for video data. We describe a warping method that can be used to augment existing architectures with very little extra computational cost. This module is called NetWarp and we demonstrate its use for a range of network architectures. The main design principle is to use optical flow of adjacent frames for warping internal network representations across time. A key insight of this work is that fast optical flow methods can be combined with many different CNN architectures for improved performance and end-to-end training. Experiments validate that the proposed approach incurs only little extra computational cost, while improving performance, when video streams are available. We achieve new state-of-the-art results on the CamVid and Cityscapes benchmark datasets and show consistent improvements over different baseline networks. Our code and models are available at http://segmentation.is.tue.mpg.de

1. Introduction

It is fair to say that the empirical performance of semantic image segmentation techniques has seen dramatic improvement in the recent years with the onset of Convolutional Neural Network (CNN) methods. The driver of this development have been large image segmentation datasets and the natural next challenge is to develop fast and accurate video segmentation methods.

The number of proposed CNN models for semantic image segmentation by far outnumbers those for video data. A naive way to use a single image CNN for video is to apply it frame-by-frame, effectively ignoring the temporal information altogether. However, frame-by-frame application often yields to jittering across frames, especially at object boundaries. Alternative approaches include the use of conditional random field (CRF) models on video data to fuse the predicted label information across frames or the development of tailored CNN architectures for videos. A separate CRF applied to the CNN predictions has the limitation, that it has no access to internal representations of the CNNs. Thus the CRF operates on a representations (the labels) that has already been condensed. Furthermore, existing CRFs for video data are often too slow for practical purposes.

We aim to develop a video segmentation technique that makes use of temporal coherence in video frames and reuse strong single image segmentation CNNs. For this, we propose a conceptually simple approach to convert existing image CNNs into video CNNs that uses only very little extra computational resources. We achieve this by 'NetWarp', a neural network module that warps the intermediate CNN representations of the previous frame to the corresponding representations of the current frame. Specifically, the NetWarp module uses the optical flow between two adjacent frames and then learns to transform the intermediate CNN representations through an extra set of operations. Multiple NetWarp modules can be used at different layers of the CNN hierarchies to warp deep intermediate representations across time, as depicted in Fig. 1.

Our implementation of NetWarp takes only about 2.5 milliseconds to process an intermediate CNN representation of 128×128 with 1024 feature channels. It is fully differentiable and can be learned using standard back propagation techniques during training of the entire CNN network. In addition, the resulting video CNN model with NetWarp modules processes the frames in an online fashion, *i.e.*, the system has access only to the present and previous frames when predicting the segmentation of the present frame.

We augmented several existing state-of-the-art image segmentation CNNs using NetWarp. On the current standard video segmentation benchmarks of CamVid [2] and Cityscapes [7], we consistently observe performance improvements in comparison to base network that is applied in a frame-by-frame mode. Our video CNNs also outperformed other recently proposed (CRF-)architectures and video propagation techniques setting up a new state-of-the-art on both CamVid and Cityscapes datasets.

In Section 2, we discuss the related works on video segmentation. In Section 3, we describe the NetWarp module and how it is used to convert image CNNs into video CNNs. In Section 4, experiments on CamVid and Cityscapes are presented. We conclude with a discussion in Section 5.

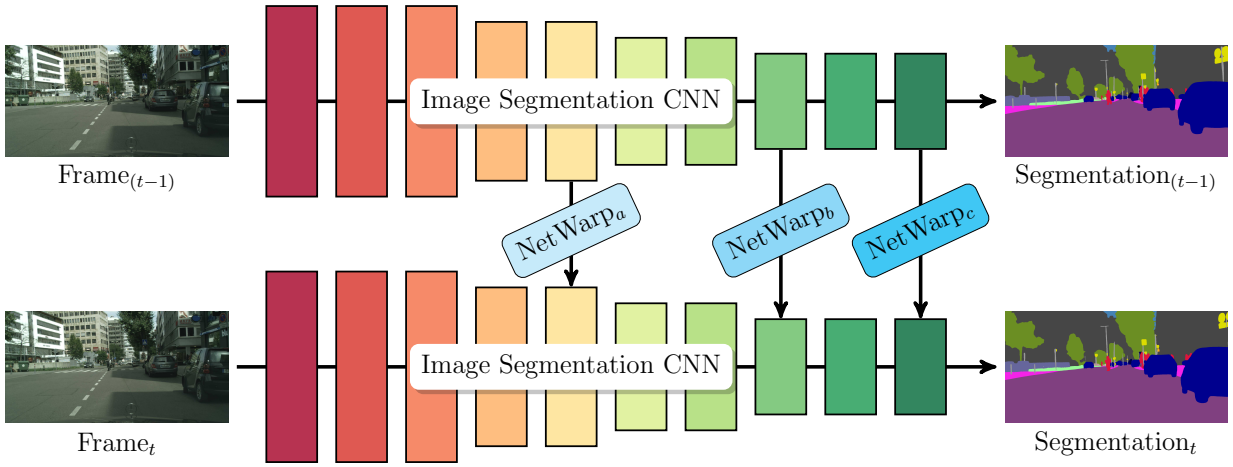

Figure 1. **Schematic of the proposed video CNN with NetWarp modules.** This illustration depicts the use of NetWarp modules in three different layers of a image CNN. The video CNN is applied in an online fashion, looking back only one frame. The CNN filter activations for the current frame are modified by the corresponding representations of the previous frame via NetWarp modules.

2. Related Works

We limit our discussion of the literature on semantic segmentation to those works concerning the video data. Most semantic video segmentation approaches implement the strategy to first obtain a single frame predictions using a classifier such as random forest or CNN, and then propagate this information using CRFs or filtering techniques to make the result temporally more consistent.

One possibility to address semantic video segmentation is by means of the 3D scene structure. Some works [3, 12, 41] build models that use 3D point clouds that have been obtained with structure from motion. Based on these geometrical and/or motion features, semantic segmentation is improved. More recent works [27, 38] propose the joint estimation of 2D semantics and 3D reconstruction of the scenes from the video data. While 3D information is very informative, it is also costly to obtain and comes with prediction errors that are hard to recover from.

A more popular route [10, 4, 8, 34, 42, 28, 32] is to construct large graphical models that connect different video pixels to achieve temporal consistency across frames. The work of [8] proposes a Perturb-and-MAP random field model with spatio-temporal energy terms based on Potts model. [4] used dynamic temporal links between the frames but optimizes for a 2D CRF with temporal energy terms. A 3D dense CRF across video frames is constructed in [42] and optimized using mean-field approximate inference. The work of [32] proposed a joint model for predicting semantic labels for supervoxels, object tracking and geometric relationship between the objects. Recently, [28] proposed a technique for optimizing the feature spaces for 3D dense CRF across video pixels. The resulting CRF model is applied on top of the unary predictions obtained with CNN or some other techniques. In [16], a joint model to estimate both optical flow and semantic segmentation is designed. [29] proposed a CRF model and an effiecient inference technique to fuse CNN unaries with long range spatio-temporal cues estimated by recurrent temporal restricted Boltzmann machine. We avoid the CRF construction and filter the intermediate CNN representations directly. This results in fast runtime and a natural way to train any augmented model by means of gradient descent.

More related to our technique are fast filtering techniques. For example, [34] learns a similarity function between pixels of consecutive frames to propagate predictions across time. The approach of [18] implements a neural network that uses learnable bilateral filters [19] for long-range propagation of information across video frames. These filtering techniques propagate information after the semantic labels are computed for each frame, whereas in contrast, our approach does filtering based propagation across intermediate CNN representations making it more integrated into CNN training.

The use of CNNs (*e.g.*, [33, 5]) resulted in a surge of performance in semantic segmentation. But, most CNN techniques work on single images. The authors of [39] observed that the semantics change slowly across time and re-use some intermediate representations from the previous frames while computing segmentation for the present frame. This results in faster runtime but a loss in accuracy. In contrast, our approach uses adjacent frame deep representations for consistent predictions across frames resulting in improved prediction accuracy.

Although several works proposed neural network approaches for processing several video frames together, they are mostly confined to video level tasks such as classifica-

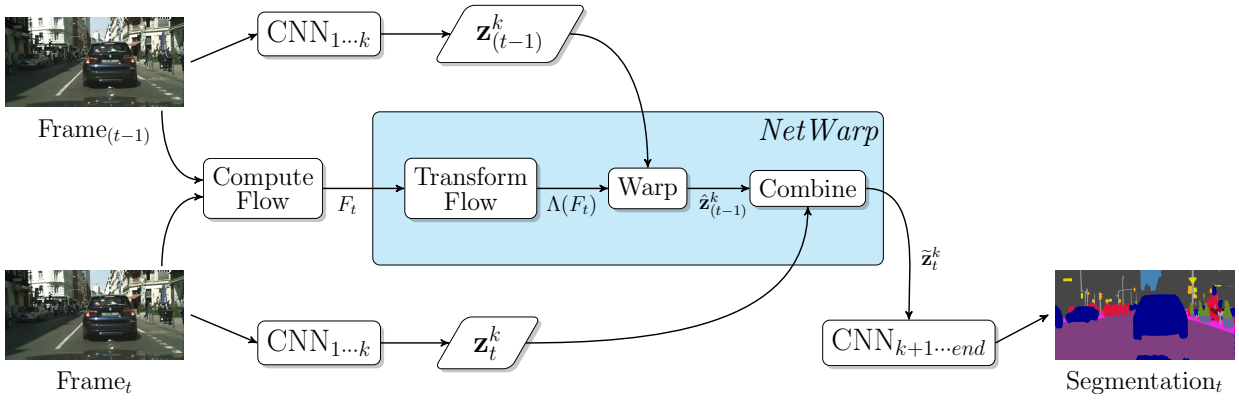

Figure 2. **Illustration of computations in a NetWarp module.** First, optical flow F_t is computed between two video frames at time steps t and $t-1$. Then the NetWarp module transforms the flow $\Lambda(F_t)$ with few convolutional layers; warps the activations $\mathbf{z}^k_{(t-1)}$ of the previous frame and and combines the warped representations with those of the present frame $\mathbf{z}^k_t$. The resulting representation $\widetilde{\mathbf{z}}^k_t$ is then passed onto the remaining CNN layers for semantic segmentation.

tion or captioning. The works of [20, 22] use 3D convolutions across frames for action recognition. In [9], LSTMs are used in a recurrent network for recognition and captioning. Two stream optical flow and image CNNs [40, 44, 32] are among the state-of-the-art approaches for visual action recognition. Unlike video level tasks, pixel-level semantic video segmentation requires filtering at pixel-level. This work proposes a way of doing local information propagation across video frames.

A related task to semantic video segmentation is video object segmentation. Like in semantic video segmentation literature, several works [36, 30, 35, 43, 17] aim to reduce the complexity of graphical model structure with spatio-temporal superpixels. Some other works use nearest neighbor fields [11] or optical flow [6] for estimating correspondence between different frame pixels. These works use pixel correspondences across frames to refine or propagate labels, whereas the proposed approach refines the intermediate CNN representations with a module that is easy to integrate into current CNN frameworks.

3. Warping Image CNNs to Video CNNs

Our aim is to convert a given CNN network architecture, designed to work on single images into a segmentation CNN for video data. Formally, given a sequence of n video frames denoted as $I_1, I_2, \cdots, I_n$, the task is to predict semantic segmentation for every video frame. Our aim is to process the video frames online, *i.e.*, the system has access only to previous frames when predicting the segmentation of the present frame.

The main building block will be the NetWarp module that warps the intermediate (k^{th} layer) CNN representations $\mathbf{z}^k_{t-1}$ of the previous frame and then combines with those of the present frame $\mathbf{z}^k_t$, where $\mathbf{z}^1_t, \mathbf{z}^2_t, \cdots, \mathbf{z}^m_t$ denote the intermediate representations of a given image CNN with m layers.

Motivation The design of the NetWarp module is based on two specific insights from the recent semantic segmentation literature. The authors of [39] showed that intermediate CNN representations change only slowly over adjacent frames, especially for deeper CNN layers. This inspired the design of the clockwork convnet architecture [39]. In [13], a bilateral inception module is constructed to average intermediate CNN representations for locations across the image that are spatially and photometrically close. There, the authors use super-pixels based on runtime considerations and demonstrated improved segmentation results when applied to different CNN architectures. Given these findings, in this work, we ask the question: *Does the combination of temporally close representations also leads to more stable and consistent semantic predictions?*

We find a positive answer to this question. Using pixel correspondences, provided by optical flow, to combine intermediate CNN representations of adjacent frames consistently improves semantic predictions for a number of CNN architectures. Especially at object boundaries and thin object structures, we observe a solid improvement. Further, this warping can be performed at different layers in CNN architectures, as illustrated in Fig. 1 and incurs only a tiny extra computation cost to the entire pipeline.

3.1. NetWarp

The NetWarp module consists of multiple separate steps, a flowchart overview is depicted in Fig. 2. It takes as input, an estimate of dense optical flow field and then performs 1. flow transformation, 2. representation warping, and 3. combination of representations. In the following, we will first discuss the optical flow computation followed by the

description of each of the three separate steps.

Flow Computation We use existing optical flow algorithms to obtain dense pixel correspondences (denoted as F_t) across frames. We chose a particular fast optical flow method to keep the runtime small. We found that DIS-Flow [26], which takes only about 5ms to compute flow per image pair (with size 640 $\times$ 480) on a CPU, works well for our purpose. Additionally, we experimented with the more accurate but slower FlowFields [1] method that requires several seconds per image pair to compute flow. Formally, given an image pair, I_t and $I_{(t-1)}$, the optical flow algorithm computes the pixel displacement (u, v) for every pixel location (x, y) in I_t to the spatial locations (x', y') in $I_{(t-1)}$. That is, $(x', y') = (x + u, y + v)$. u and v are floating point numbers and represent pixel displacements in horizontal and vertical directions respectively. Note that we compute the *reverse flow* mapping the present frame locations to locations in previous frame as we want to warp previous frame representations.

Flow Transformation Correspondences obtained with traditional optical flow methods might not be optimal for propagating representations across video frames. So, we use a small CNN to transform the pre-computed optical flow, which we will refer to as FlowCNN and denote the transformation as $\Lambda(F_t)$. We concatenate the original two channel flow, the previous and present frame images, and the difference of the two frames. This results in a 11 channel tensor as an input to the FlowCNN. The network itself is composed of 4 convolution layers interleaved with ReLU nonlinearities. All the convolution layers are made up of 3 $\times$ 3 filters and the number of output channels for the first 3 layers are 16, 32 and 2 respectively. The output of the third layer is then concatenated (skip connection) with the original pre-computed flow which is then passed on to the last 3 $\times$ 3 convolution layer to obtain final transformed flow. This network architecture is loosely inspired from the residual blocks in ResNet [15] architectures. Other network architectures are conceivable. All parameters of FlowCNN are learned via standard back-propagation. Learning is done on semantic segmentation only and we do not include any supervised flow data as we are mainly interested in semantic video segmentation. Figure 5 in the experimental section shows how the flow transforms with the FlowCNN. We observe significant transformations in the original flow with the FlowCNN and we will discuss more about these changes in the experimental section.

Warping Representations The FlowCNN transforms a dense correspondence field from frame I_t to the previous frame $I_{(t-1)}$. Let us assume that we want to apply the NetWarp module on the k^{th} layer of the image CNN and the filter activations for the adjacent frames are $\mathbf{z}_t^k$ and $\mathbf{z}_{(t-1)}^k$ (as in Fig 2). For notational convenience, we drop k and refer to these as $\mathbf{z}_t$ and $\mathbf{z}_{(t-1)}$ respectively. The representations of the previous frame $\mathbf{z}_{(t-1)}$ are warped to align with the corresponding present frame representations:

$$\hat{\mathbf{z}}_{(t-1)} = Warp(\mathbf{z}_{(t-1)}, \Lambda(F_t)), \tag{1}$$

where $\hat{\mathbf{z}}_{(t-1)}$ denotes the warped representations, F_t is the dense correspondence field and $\Lambda(\cdot)$ represents the FlowCNN described above. Lets say we want to compute the warped representations $\hat{\mathbf{z}}_{(t-1)}$ at a present frame's pixel location (x, y) which is mapped to the location (x', y') in the previous frame by the transformed flow. We implement $Warp$ as a bilinear interpolation of $\mathbf{z}_{(t-1)}$ at the desired points (x', y'). Let (x_1, y_1), (x_1, y_2), (x_2, y_1) and (x_2, y_2) be the corner points of the previous frame's grid cell where (x', y') falls. Then the warping of $\mathbf{z}_{(t-1)}$ to obtain $\hat{\mathbf{z}}_{(t-1)}(x, y)$ is given as standard bilinear interpolation:

$$\begin{aligned}\hat{\mathbf{z}}_{(t-1)}(x, y) &= \mathbf{z}_{(t-1)}(x', y') \\ &= \frac{1}{\eta} \begin{bmatrix} x_2 - x' \\ x' - x_1 \end{bmatrix}^\top \begin{bmatrix} \mathbf{z}_{(t-1)}(x_1, y_1) & \mathbf{z}_{(t-1)}(x_1, y_2) \\ \mathbf{z}_{(t-1)}(x_2, y_1) & \mathbf{z}_{(t-1)}(x_2, y_2) \end{bmatrix} \begin{bmatrix} y_2 - y' \\ y' - y_1 \end{bmatrix}\end{aligned} \tag{2}$$

where $\eta = 1/(x_2 - x_1)(y_2 - y_1)$. In case (x', y') lies outside the spatial domain of $\mathbf{z}_{(t-1)}$, we back-project (x', y') to the nearest border in $\mathbf{z}_{(t-1)}$. The above warping function is differentiable at all the points except when the flow values are integer numbers. Intuitively, this is because the the corner points used for the interpolation suddenly change when (x', y') moves across from one grid cell to another. To avoid the non-differentiable case, we add a small ϵ of 0.0001 to the transformed flow. This makes the warping module differentiable with respect to both the previous frame representations and the transformed flow. We implement gradients using standard matrix calculus. Due to strided pooling, deeper CNN representations are typically of smaller resolution in comparison to the image signal. The same strides are used for the transformed optical flow to obtain the pixel correspondences at the desired resolution.

Combination of Representations Once the warped activations of the previous frame $\hat{\mathbf{z}}_{(t-1)}^k$ are computed with the above mentioned procedure, they are linearly combined with the present frame representations $\mathbf{z}_t^k$

$$\widetilde{\mathbf{z}}_t^k = \mathbf{w}_1 \odot \mathbf{z}_t^k + \mathbf{w}_2 \odot \hat{\mathbf{z}}_{(t-1)}^k, \tag{3}$$

where $\mathbf{w}_1$ and $\mathbf{w}_2$ are weight vectors with the same length as the number of channels in $\mathbf{z}^k$; and $\odot$ represents per-channel scalar multiplication. In other words, each channel of the frame t and the corresponding channel of the warped representations in the previous frame $t-1$ are linearly combined. The parameters $\mathbf{w}_1, \mathbf{w}_2$ are learned via standard back-propagation. The result $\widetilde{\mathbf{z}}_t^k$ is then passed on to the remaining image CNN layers. Different computations in the NetWarp module are illustrated in Fig. 2.

PlayData-CNN (IoU: 68.9)	Conv1_2	Conv2_2	Conv3_3	Conv4_3	Conv5_3	FC6	FC7	FC8	FC6 + FC7
+ NetWarp (without FlowCNN)	69.3	69.5	69.5	69.4	69.5	69.6	69.4	69.3	**69.8**
+ NetWarp (with FlowCNN)	69.6	69.6	69.6	69.5	69.7	69.8	69.7	69.5	**70.2**

Table 1. **The effect of where NetWarp modules are inserted.** Shown are test IoU scores on CamVid for augmented versions of the PlayData-CNN. We observe an improvement (frame-by-frame results 68.9) independent of where a NetWarp is included. Refining the flow estimate typically leads to slightly better results.

Usage and Training The inclusion of NetWarp modules still allows end-to-end training. It can be easily integrated in different deep learning architectures. Note that back-propagating a loss from frame t will affect image CNN layers (those preceding NetWarp modules) for the present and also previous frames. We use shared filter weights for the image CNN across the frames. Training is possible also when ground truth label information is available for only some and not all frames, which is generally the case.

Due to GPU memory constraints, we make an approximation and only use two frames at a time. Filter activations from frame $t-1$ would receive updates from $t-2$ when unrolling the architecture in time, but we ignore this effect because of the hardware memory limitations. The NetWarp module can be included at different depths and multiple NetWarp modules can be used to form a video CNN. In our experiments, we used the same flow transformation $\Lambda(\cdot)$ when multiple NetWarp modules are used. We used the Adam [23] stochastic gradient descent method for optimizing the network parameters. Combination weights are initialized with $\mathbf{w}_1 = 1$ and $\mathbf{w}_2 = 0$, so the initial video network is identical to the single image CNN. Other NetWarp parameters are initialized randomly with a Gaussian noise. All our experiments and runtime analysis were performed using a Nvidia TitanX GPU and a 6 core Intel i7-5820K CPU clocked at 3.30GHz machine. Our implementation that builds on the Caffe [21] framework is available at http://segmentation.is.tue.mpg.de.

4. Experiments

We evaluated the NetWarp modules using the two challenging semantic video segmentation benchmarks for which video frames and/or annotations are available: CamVid [2] and Cityscapes [7]. Both datasets contain real world video sequences of street scenes. We choose different popular CNN architectures of [47, 37, 48] and augmented them with the NetWarp modules at different places across the network. We follow standard protocols and report the standard Intersection over Union (IoU) score which is defined in terms of true-positives (TP), false-positives (FP) and false-negatives (FP): "TP / (TP + FP + FN)" and additionally the instance-level iIoU for Cityscapes [7]. We are particularly interested in the segmentation effects around the boundaries. Methods like the influential DenseCRFs [25] are particularly good in segmentation of the object boundaries. Therefore, we adopt the methodologies from [24, 25] and measure the IoU performance only in a narrow band around the ground truth label changes (see Fig.17 in [24]). We vary the width of this band and refer to this measure as trimap-IoU (tIoU). In all the experiments, unless specified, we use a default trimap band of 2 pixels.

4.1. CamVid Dataset

The CamVid dataset contains 4 videos with ground-truth labelling available for every 30th frame. Overall, the dataset has 701 frames with ground-truth. For direct comparisons with previous works, we used the same train, validation and test splits as in [47, 28, 37]. In all our models, we use only the *train* split for training and report the performance on the *test* split. We introduced NetWarp modules in two popular segmentation CNNs for this dataset: One is PlayData-CNN from [37] and another is Dilation-CNN from [47]. Unless otherwise mentioned, we used DIS-Flow [26] for the experiments on this dataset.

With PlayData-CNN [37] as the base network, we first study how the NetWarp module performs when introduced at different stages of the network. The network architecture of PlayData-CNN is made of five convolutional blocks, each with 2 or 3 convolutional layers, followed by three 1×1 convolution layers (FC layers). We add the NetWarp module to the following layers at various depths of the network: $Conv_{1\_2}$, $Conv_{2\_2}$, $Conv_{3\_3}$, $Conv_{4\_3}$, $Conv_{5\_3}$, FC_6, FC_7 and FC_8 layers. The corresponding IoU scores are reported in Tab. 1. We find a consistent improvement over the PlayData-CNN performance of 68.9% irrespective of the NetWarp locations in the network. Since NetWarp modules at FC_6 and FC_7 performed slightly better, we chose to insert NetWarp at both the locations in our final model with this base network. We also observe consistent increase in IoU with the use of flow transformation across different NetWarp locations. Our best model with two NetWarp modules yields an IoU score of 70.2% which is a new state-of-the-art on this dataset.

Adding NetWarp modules to CNN introduces very few additional parameters. For example, two NetWarp modules at FC_6 and FC_7 in the PlayData-CNN have about 16K parameters, a mere 0.012% of all 134.3M parameters. The experiments in Tab. 1 indicate that improvements can be contributed to the temporal information propagation at multiple-depths. As a baseline, concatenating corresponding FC_6 and FC_7 features resulted in only 0.1% IoU improvement compared to 1.3% using NetWarp modules.

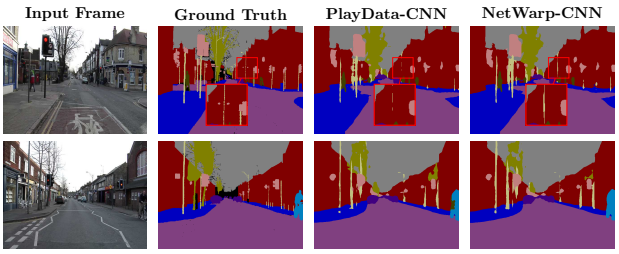

Figure 3. **Qualitative results from the CamVid dataset.** Notice how NetWarp-CNN recovers some thin structures in the top row and corrects some regions (on cyclist) in the second row.

	IoU	tIoU
PlayData-CNN [37]	68.9	39.0
+ NetWarp (with DIS-Flow [26])	70.2	39.9
+ NetWarp (with FlowFields [1])	**70.3**	**40.1**
+ VPN [18]	69.5	-

Table 2. **CamVid Results using PlayData-CNN.** Shown are the IoU and tIoU scores from different methods using a fast flow from DIS[26] and an accurate flow from [1] for NetWarp augmented PlayData-CNN.

	IoU	tIoU	Runtime (ms)
Dilation-CNN [47]	65.3	34.7	380
+ NetWarp(Ours)	**67.1**	**36.6**	395
+ FSO-CRF [28]	66.1	-	$>$ 10k
+ VPN [18]	66.7	36.1	680

Table 3. **CamVid Results using Dilation-CNN.** IoU, tIoU scores and runtimes (in milliseconds) for different methods.

In Tab. 2, we show the effect of using the more accurate but slower optical flow method of Flowfields [1]. Results indicate that there is only a 0.1% improvement in IoU with Flowfields but this incurs a much higher runtime for flow computation. Results also indicate that our approach outperformed the current state-of-the-art approach of VPN from [18] by a significant margin of 0.8% IoU, while being faster.

As a second network, we choose the dilation CNN from [47]. This network consists of a standard CNN followed by a context module with 8 dilated convolutional layers. For this network, we apply the NetWarp module on the output of each of these 8 dilated convolutions. Table 3 shows the performance and runtime comparisons with the dilation CNN and other related techniques. With a runtime increase of 15 milliseconds, we observe significant improvements in the order of 1.8% IoU. The runtime increase assumes that the result of the previous frame is already computed, which is the case for video segmentation.

4.2. Cityscapes Dataset

The Cityscapes dataset [7] comes with a total of 5000 video sequences of high-quality images (2048×1024 resolution), partitioned into 2975 train, 500 validation and 1525 test sequences. The videos are captured in different weather conditions across 50 different cities in Germany and Switzerland. In addition to the IoU and tIoU performance metrics, we report the instance-level IoU score. Since IoU score is dominated by large objects/regions (such as road) in the scene, the makers of this dataset proposed the iIoU score that takes into account the relative size of different objects/regions. The iIoU score is given as iTP/(iTP + FP + iFN), where iTP and iFN are the modified true-positive and false-negative scores which are computed by weighting the contribution of each pixel by the ratio of the average class instance size to the size of the respective ground truth instance. This measures how well the semantic labelling represents the individual instances in the scene. For more details on this metric, please refer to the original dataset paper [7]. For this dataset, we used DIS-Flow [26] for all networks augmented with NetWarp modules.

We choose the recently proposed Pyramid Scene Parsing Network (PSPNet) from [48]. Because of high-resolution images in Cityscapes and GPU memory limitations, PSPNet is applied in a sliding window fashion with a window size of 713×713. To achieve higher segmentation performance, the authors of [48] also evaluated a multi-scale version. Applying the *same* PSPNet on 6 different scales of an input image results to an improvement of 1.4% IoU over the single-scale variant. This increased performance comes at the cost of increased runtime. In the single-scale setting, the network is evaluated on 8 different windows to get a full image result, whereas in the multi-scale setting, the network is evaluated 81 times leading to 10 times increase in runtime. We refer to the single-scale and multi-scale evaluations as PSPNet-SSc and PSPNet-MSc respectively.

The architecture of PSPNet is a ResNet101 [15] network variant with pyramid style pooling layers. We insert NetWarp modules before and after the pyramid pooling layers. More precisely, NetWarp modules are added on both the $Conv_{4\_23}$ and $Conv_{5\_4}$ representations. The network is then fine-tuned with 2975 train video sequences without any data augmentation. We evaluate both the single-scale and multi-scale variants. Table 4 summarizes the quantitative results with and without NetWarp modules, on validation data scenes of Cityscapes. We find an improvement of the IoU (by 1.2), tIoU (by 2.4) and iIoU (by 1.4) respectively over the single image PSPNet-SSc variant. These improvements come with a low runtime overhead of 24 milliseconds. Also the augmented multi-scale network improves all measures: IoU by 0.7, tIoU by 1.8, and iIoU by 1.4%.

We chose to submit the results of the best performing method from the validation set to the Cityscapes test server. Results are shown in Tab. 5. We find that the NetWarp augmented PSP multi-scale variant is on par with the current top performing method [46] (80.5 vs. 80.6) and out-performs current top models in terms of iIoU by a sig-

	IoU	iIoU	tIoU	Runtime (s)
PSPNet-SSc [48]	79.4	60.7	39.7	3.00
+NetWarp	**80.6**	**62.1**	**42.1**	3.04
PSPNet-MSc [48]	80.8	62.2	41.5	30.3
+NetWarp	**81.5**	**63.6**	**43.3**	30.5

Table 4. **Performance of PSPNet with NetWarp modules on the Cityscapes validation dataset.** We observe consistent improvements with NetWarp modules across all three performance measures in both the single-scale (SSc) and multi-scale (MSc) settings, while only adding little time overhead.

nificant margin of 1.4%. In summary, at submission time, our result is best performing method in iIoU and second on IoU[1]. Surprisingly, it is the only approach that uses video data. A closer look into class IoU score in Tab. 5 shows that our approach works particularly well for parsing thin structures like poles, traffic lights, traffic signs etc. The improvement is around 4% IoU for the pole class. Another observation is that adding NetWarp modules resulted in slight IoU performance decrease for few broad object classes such as car, truck etc. However, we find consistent improvements across most of the classes in other performance measures. The classwise iIoU scores that are computed on broad object classes like car, bus etc show better performance on 7 out of 8 classes for NetWarp. Further, analysing the classwise tIoU scores on the validation set, we find that NetWarpperforms better on 17 out of 19 classes. Visual results in Fig. 6 also indicate that the thin structures are better parsed with the introduction of NetWarp modules. Qualitatively, we find improved performance near boundaries compared to baseline CNN (see supplementary video). Visual results in Fig. 6 also indicate that NetWarp helps in rectifying the segmentation of big regions as well.

In Fig. 4, we show the relative improvement of the NetWarp augmented PSPNet for different widths in the trimap-IoU. From this analysis we can conclude that the IoU improvement is especially due to better performance near object boundary. This is true for both the single and the multi-scale version of the network. Image CNNs, in general, are very good at segmenting large regions or objects like road or cars. However, thin and fine structures are difficult to parse as information is lost due to strided operations inside CNN. In part this is recovered by NetWarp CNNs that use the temporal context to recover thin structures. In Fig. 6, some qualitative results with static image CNN and our video CNN are shown. We observe that the NetWarp module correct for thin structures but also frequently correct larger regions of wrong segmentations. This is possible since representations for the same regions are combined over different frames leading to a more robust classification.

Next, we analyze how the DIS-Flow is transformed by the FlowCNN. Figure 5 shows some visualizations of the

[1] https://www.cityscapes-dataset.com/benchmarks/

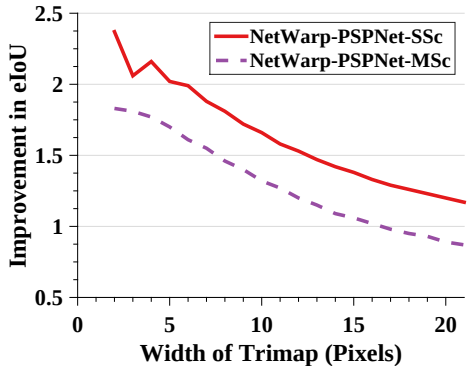

Figure 4. **tIoU improvement.** Relative improvements of IoU within trimaps as a function of trimap width. Most improvement is in regions close to object boundaries.

Figure 5. **Effect of flow transformation.** Example results of input and transformed flow, after training for semantic segmentation. There is a qualitative difference between CamVid and CityScapes. Best viewed on screen.

transformed flow along with the original DIS Flow fields. We can clearly observe that, in both CamVid and Cityscapes images, the structure of the scene is much more pronounced in the transformed flow indicating that the traditionally computed flow might not be optimal to find pixel correspondences for semantic segmentation.

5. Conclusions and Outlook

We presented NetWarp, an efficient and conceptually easy way to transform image CNNs into video CNNs. The main concept is to transfer intermediate CNN filter activ-

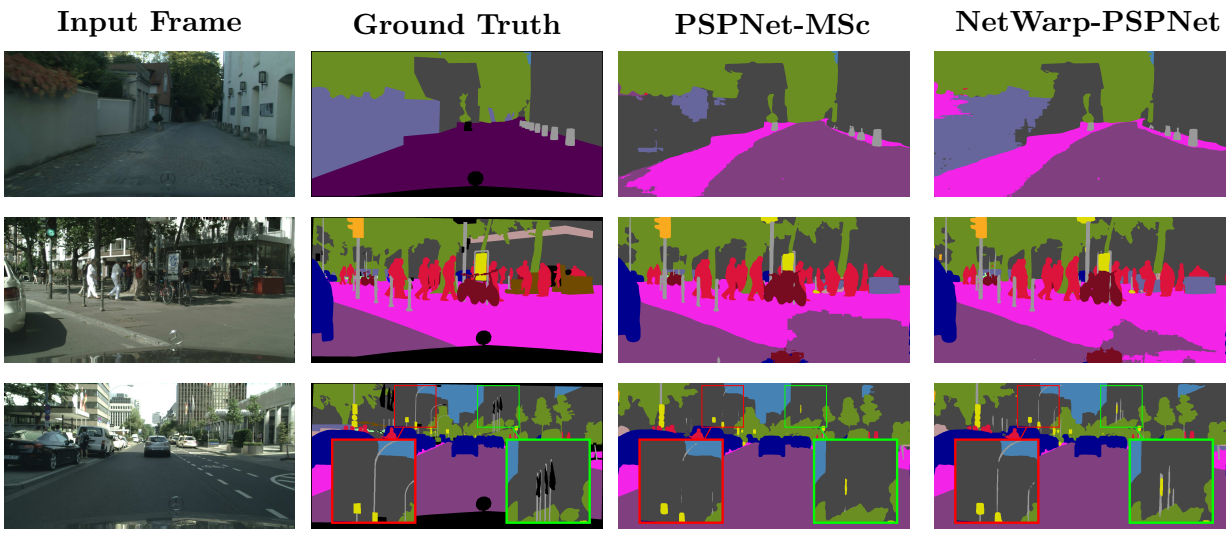

Figure 6. **Qualitative results from the Cityscapes dataset.** Observe how NetWarp-PSPNet is able to correct larger parts of wrong segmentation (top two rows) by warping activations across frames. The bottom row shows an example of improved segmentation of a thin structure. All results shown are obtained with the multi-scale versions.

	iIoU	IoU	road	sidewalk	building	wall	fence	pole	traffic light	traffic sign	vegetation	terrain	sky	person	rider	car	truck	bus	train	motorcycle	bicycle
PSPNet-MSc [48]	58.1	80.2	98.6	86.6	93.2	58.1	63.0	64.5	75.2	79.2	93.4	72.1	95.1	86.3	71.4	96.0	73.6	90.4	80.4	69.9	76.9
+NetWarp(Ours)	**59.5**	80.5	98.6	86.7	93.4	60.6	62.6	68.6	75.9	80.0	93.5	72.0	95.3	86.5	72.1	95.9	72.9	90.0	77.4	70.5	76.4
ResNet-38 [46]	57.8	**80.6**	98.7	87.0	93.3	60.4	62.9	67.6	75.0	78.7	93.7	73.7	95.5	86.8	71.1	96.1	75.2	87.6	81.9	69.8	76.7
TuSimple [45]	56.9	80.1	98.6	85.9	93.2	57.7	61.2	67.2	73.7	78.0	93.4	72.3	95.4	85.9	70.5	95.9	76.1	90.6	83.7	67.4	75.7
LRR-4X [14]	47.9	71.9	98.0	81.5	91.4	50.5	52.7	59.4	66.8	72.7	92.5	70.1	95.0	81.3	60.1	94.3	51.2	67.7	54.6	55.6	69.6
RefineNet [31]	47.2	73.6	98.2	83.3	91.3	47.8	50.4	56.1	66.9	71.3	92.3	70.3	94.8	80.9	63.3	94.5	64.6	76.1	64.3	62.2	70.0

Table 5. **Results on the Cityscapes test dataset.** Average iIoU, IoU and class IoU scores of base PSPNet, with NetWarp modules and also other top performing methods taken from the Cityscapes benchmark website at the time of submission. The NetWarp augmented PSPNet-MSc version achieves highest iIoU and is about on par with [46] on IoU. Our video methods performs particularly well on small classes such as poles, traffic lights etc.

ities of adjacent frames based on transformed optical flow estimate. The resulting video CNN is end-to-end trainable, runs in an online fashion and has only a small computation overhead in comparison to the frame-by-frame application. Experiments on the current standard benchmark datasets CityScapes and CamVid show improved performance for several strong baseline methods. The final model sets a new state-of-the-art performance on both CityScapes and CamVid.

Extensive experimental analysis provide insights into the workings of the NetWarp module. First, we demonstrate consistent performance improvements across different image CNN hierarchies. Second, we find more temporally consistent semantic predictions and better coverage of thin structures such as poles and traffic signs. Third, we observed that the flow changed radically after the transformation (FlowCNN) trained with the segmentation loss. From the qualitative results, it seems that the optical flow at the object boundaries is important for semantic segmentation. An interesting future work is to systematically study what properties of optical flow estimation are necessary for semantic segmentation and the impact of different types of interpolation in a NetWarp module.

Another future direction is to scale the video CNNs to use multiple frames. Due to GPU memory limitations and to keep training easier, here, we trained with only two adjacent frames at a time. In part this is due to the memory demanding base models like ResNet101. Memory optimizing the CNN training would alleviate some of the problems and enables training with many frames together. We also envision that the findings of this paper are interesting for the design of video CNNs for different tasks other than semantic segmentation.

Acknowledgements. This work was supported by Max Planck ETH Center for Learning Systems and the research program of the Bernstein Center for Computational Neuroscience, Tübingen, funded by the German Federal Ministry of Education and Research (BMBF; FKZ: 01GQ1002). We thank Thomas Nestmeyer and Laura Sevilla for help with the supplementary.

References

[1] C. Bailer, B. Taetz, and D. Stricker. Flow fields: Dense correspondence fields for highly accurate large displacement optical flow estimation. In *Proceedings of the IEEE International Conference on Computer Vision*, 2015. 4, 6

[2] G. J. Brostow, J. Fauqueur, and R. Cipolla. Semantic object classes in video: A high-definition ground truth database. *Pattern Recognition Letters*, 30(2):88–97, 2009. 1, 5

[3] G. J. Brostow, J. Shotton, J. Fauqueur, and R. Cipolla. Segmentation and recognition using structure from motion point clouds. In *European Conference on Computer Vision*, pages 44–57. Springer, 2008. 2

[4] A. Y. Chen and J. J. Corso. Temporally consistent multi-class video-object segmentation with the video graph-shifts algorithm. In *IEEE Winter Conference on Applications of Computer Vision*, pages 614–621. IEEE, 2011. 2

[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. *arXiv preprint arXiv:1412.7062*, 2014. 2

[6] Y.-Y. Chuang, A. Agarwala, B. Curless, D. H. Salesin, and R. Szeliski. Video matting of complex scenes. *ACM Transactions on Graphics (ToG)*, 21(3):243–248, 2002. 3

[7] M. Cordts, M. Omran, S. Ramos, T. Rehfeld, M. Enzweiler, R. Benenson, U. Franke, S. Roth, and B. Schiele. The cityscapes dataset for semantic urban scene understanding. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 3213–3223, 2016. 1, 5, 6

[8] R. de Nijs, S. Ramos, G. Roig, X. Boix, L. Van Gool, and K. Kühnlenz. On-line semantic perception using uncertainty. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 4185–4191. IEEE, 2012. 2

[9] J. Donahue, L. Anne Hendricks, S. Guadarrama, M. Rohrbach, S. Venugopalan, K. Saenko, and T. Darrell. Long-term recurrent convolutional networks for visual recognition and description. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2625–2634, 2015. 3

[10] A. Ess, T. Mueller, H. Grabner, and L. J. Van Gool. Segmentation-based urban traffic scene understanding. In *British Machine Vision Conference*, 2009. 2

[11] Q. Fan, F. Zhong, D. Lischinski, D. Cohen-Or, and B. Chen. Jumpcut: non-successive mask transfer and interpolation for video cutout. *ACM Transactions on Graphics (ToG)*, 34(6):195, 2015. 3

[12] G. Floros and B. Leibe. Joint 2d-3d temporally consistent semantic segmentation of street scenes. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 2823–2830, 2012. 2

[13] R. Gadde, V. Jampani, M. Kiefel, D. Kappler, and P. Gehler. Superpixel convolutional networks using bilateral inceptions. In *European Conference on Computer Vision*. Springer, 2016. 3

[14] G. Ghiasi and C. C. Fowlkes. Laplacian pyramid reconstruction and refinement for semantic segmentation. In *European Conference on Computer Vision*, pages 519–534. Springer, 2016. 8

[15] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 770–778, 2016. 4, 6

[16] J. Hur and S. Roth. Joint optical flow and temporally consistent semantic segmentation. In *European Conference on Computer Vision*, pages 163–177. Springer, 2016. 2

[17] S. D. Jain and K. Grauman. Supervoxel-consistent foreground propagation in video. In *European Conference on Computer Vision*, pages 656–671. Springer, 2014. 3

[18] V. Jampani, R. Gadde, and P. V. Gehler. Video propagation networks. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2017. 2, 6

[19] V. Jampani, M. Kiefel, and P. V. Gehler. Learning sparse high dimensional filters: Image filtering, dense crfs and bilateral neural networks. In *IEEE Conference on Computer Vision and Pattern Recognition*, June 2016. 2

[20] S. Ji, W. Xu, M. Yang, and K. Yu. 3d convolutional neural networks for human action recognition. *IEEE transactions on pattern analysis and machine intelligence*, 35(1):221–231, 2013. 3

[21] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell. Caffe: Convolutional architecture for fast feature embedding. In *Proceedings of the ACM International Conference on Multimedia*, pages 675–678. ACM, 2014. 5

[22] A. Karpathy, G. Toderici, S. Shetty, T. Leung, R. Sukthankar, and L. Fei-Fei. Large-scale video classification with convolutional neural networks. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 1725–1732, 2014. 3

[23] D. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. 5

[24] P. Kohli, P. H. Torr, et al. Robust higher order potentials for enforcing label consistency. *International Journal of Computer Vision*, 82(3):302–324, 2009. 5

[25] P. Krähenbühl and V. Koltun. Efficient inference in fully connected crfs with gaussian edge potentials. In *Advances in neural information processing systems*, 2011. 5

[26] T. Kroeger, R. Timofte, D. Dai, and L. V. Gool. Fast optical flow using dense inverse search. In *European Conference on Computer Vision*, pages 471–488. Springer, 2016. 4, 5, 6

[27] A. Kundu, Y. Li, F. Dellaert, F. Li, and J. M. Rehg. Joint semantic segmentation and 3d reconstruction from monocular video. In *European Conference on Computer Vision*, pages 703–718. Springer, 2014. 2

[28] A. Kundu, V. Vineet, and V. Koltun. Feature space optimization for semantic video segmentation. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 3168–3175, 2016. 2, 5, 6

[29] P. Lei and S. Todorovic. Recurrent temporal deep field for semantic video labeling. In *European Conference on Computer Vision*, pages 302–317. Springer, 2016. 2

[30] Y. Li, J. Sun, and H.-Y. Shum. Video object cut and paste. *ACM Transactions on Graphics (ToG)*, 24(3):595–600, 2005. 3

[31] G. Lin, A. Milan, C. Shen, and I. Reid. RefineNet: Multi-path refinement networks for high-resolution semantic segmentation. In *CVPR*, 2017. 8

[32] B. Liu, X. He, and S. Gould. Multi-class semantic video segmentation with exemplar-based object reasoning. In *Applications of Computer Vision (WACV), 2015 IEEE Winter Conference on*, pages 1014–1021. IEEE, 2015. 2, 3

[33] J. Long, E. Shelhamer, and T. Darrell. Fully convolutional networks for semantic segmentation. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 3431–3440, 2015. 2

[34] O. Miksik, D. Munoz, J. A. Bagnell, and M. Hebert. Efficient temporal consistency for streaming video scene analysis. In *Robotics and Automation (ICRA), 2013 IEEE International Conference on*, pages 133–139. IEEE, 2013. 2

[35] B. L. Price, B. S. Morse, and S. Cohen. Livecut: Learning-based interactive video segmentation by evaluation of multiple propagated cues. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 779–786. IEEE, 2009. 3

[36] M. Reso, B. Scheuermann, J. Jachalsky, B. Rosenhahn, and J. Ostermann. Interactive segmentation of high-resolution video content using temporally coherent superpixels and graph cut. In *International Symposium on Visual Computing*, pages 281–292. Springer, 2014. 3

[37] S. R. Richter, V. Vineet, S. Roth, and V. Koltun. Playing for data: Ground truth from computer games. In *European Conference on Computer Vision*, pages 102–118. Springer, 2016. 5, 6

[38] S. Sengupta, E. Greveson, A. Shahrokni, and P. H. Torr. Urban 3d semantic modelling using stereo vision. In *Robotics and Automation (ICRA), 2013 IEEE International Conference on*, pages 580–585. IEEE, 2013. 2

[39] E. Shelhamer, K. Rakelly, J. Hoffman, and T. Darrell. Clockwork convnets for video semantic segmentation. *arXiv preprint arXiv:1608.03609*, 2016. 2, 3

[40] K. Simonyan and A. Zisserman. Two-stream convolutional networks for action recognition in videos. In *Advances in neural information processing systems*, pages 568–576, 2014. 3

[41] P. Sturgess, K. Alahari, L. Ladicky, and P. H. Torr. Combining appearance and structure from motion features for road scene understanding. In *British Machine Vision Conference*. BMVA, 2009. 2

[42] S. Tripathi, S. Belongie, Y. Hwang, and T. Nguyen. Semantic video segmentation: Exploring inference efficiency. In *2015 International SoC Design Conference (ISOCC)*, pages 157–158. IEEE, 2015. 2

[43] J. Wang, P. Bhat, R. A. Colburn, M. Agrawala, and M. F. Cohen. Interactive video cutout. *ACM Transactions on Graphics (ToG)*, 24(3):585–594, 2005. 3

[44] L. Wang, Y. Xiong, Z. Wang, Y. Qiao, D. Lin, X. Tang, and L. Van Gool. Temporal segment networks: towards good practices for deep action recognition. In *European Conference on Computer Vision*, pages 20–36. Springer, 2016. 3

[45] P. Wang, P. Chen, Y. Yuan, D. Liu, Z. Huang, X. Hou, and G. Cottrell. Understanding convolution for semantic segmentation. *arXiv preprint arXiv:1702.08502*, 2017. 8

[46] Z. Wu, C. Shen, and A. v. d. Hengel. Wider or deeper: Revisiting the resnet model for visual recognition. *arXiv preprint arXiv:1611.10080*, 2016. 6, 8

[47] F. Yu and V. Koltun. Multi-scale context aggregation by dilated convolutions. In *International Conference on Learning Representations*, 2016. 5, 6

[48] H. Zhao, J. Shi, X. Qi, X. Wang, and J. Jia. Pyramid scene parsing network. *arXiv preprint arXiv:1612.01105*, 2016. 5, 6, 7, 8