Models Matter, So Does Training: An Empirical Study of CNNs for Optical Flow Estimation

Deqing Sun, Xiaodong Yang, Ming-Yu Liu, and Jan Kautz

Abstract—We investigate two crucial and closely related aspects of CNNs for optical flow estimation: models and training. First, we design a compact but effective CNN model, called PWC-Net, according to simple and well-established principles: pyramidal processing, warping, and cost volume processing. PWC-Net is 17 times smaller in size, 2 times faster in inference, and 11% more accurate on Sintel final than the recent FlowNet2 model. It is the winning entry in the optical flow competition of the robust vision challenge. Next, we experimentally analyze the sources of our performance gains. In particular, we use the same training procedure of PWC-Net to retrain FlowNetC, a sub-network of FlowNet2. The retrained FlowNetC is 56% more accurate on Sintel final than the previously trained one and even 5% more accurate than the FlowNet2 model. We further improve the training procedure and increase the accuracy of PWC-Net on Sintel by 10% and on KITTI 2012 and 2015 by 20%. Our newly trained model parameters and training protocols will be available on https://github.com/NVlabs/PWC-Net.

Index Terms—Optical flow, pyramid, warping, cost volume, and convolutional neural network (CNN).

1 Introduction

Models matter. Since the seminal work of AlexNet [1] demonstrated the advantages of deep models over shallow ones on the ImageNet challenge [2], many novel deep convolutional neural network (CNN) [3] models have been proposed and have significantly improved in performance, such as VGG [4], Inception [5], ResNet [6], and DenseNet [7]. Fast, scalable, and end-to-end trainable CNNs have significantly advanced the field of computer vision in recent years, and particularly high-level vision problems.

Inspired by the successes of deep learning in high-level vision tasks, Dosovitskiy *et al.* [8] propose two CNN models for optical flow, *i.e.*, FlowNetS and FlowNetC, and introduce a paradigm shift to this fundamental low/middle-level vision problem. Their work shows the feasibility of directly estimating optical flow from raw images using a generic U-Net CNN architecture [9]. Although their performance is below the state of the art, FlowNetS and FlowNetC are the best among their contemporary real-time methods.

Recently, Ilg *et al.* [10] stacked one FlowNetC and several FlowNetS networks into a large model, called FlowNet2, which performs on par with state-of-the-art methods but runs much faster (Fig. 1). However, large models are more prone to the over-fitting problem, and as a result, the sub-networks of FlowNet2 have to be trained sequentially. Furthermore, FlowNet2 requires a memory footprint of 640MB and is not well-suited for mobile and embedded devices.

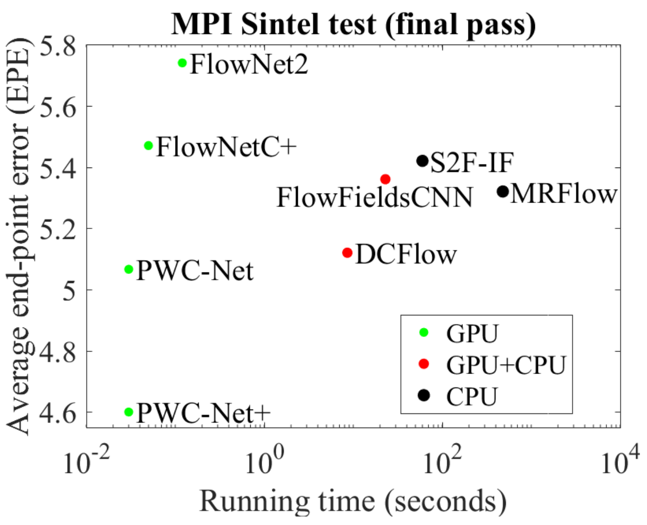
SpyNet [11] addresses the model size issue by combining deep learning with two classical optical flow estimation principles. SpyNet uses a spatial pyramid network and warps the second image toward the first one using the initial flow. The motion between the first and warped images is usually small. Thus SpyNet only needs a small network to estimate the motion from these two images. SpyNet performs on par with FlowNetC but below FlowNetS and FlowNet2. The reported results by FlowNet2 and SpyNet show a clear trade-off between accuracy and model size.

Is it possible to both increase the accuracy and reduce the size of a CNN model for optical flow estimation? In principle, the trade-off between model size and accuracy imposes a fundamental limit for general machine learning algorithms. However, we find that combining domain knowledge with deep learning can achieve both goals simultaneously for the particular problem of optical flow estimation.

SpyNet shows the potential of combining classical principles with CNNs. However, we argue that its performance gap with FlowNetS and FlowNet2 is due to the partial use of classical principles. First, traditional optical flow methods often pre-process the raw images to extract features that are invariant to shadows or lighting changes [12], [13]. Further, in the special case of stereo matching, a cost volume is a more discriminative representation of the disparity (1D flow) than raw images or features [14], [15], [16]. While constructing a full cost volume is computationally prohibitive for real-time optical flow estimation [17], our work constructs a "partial" cost volume by limiting the search range at each pyramid level. Linking different pyramid levels using a warping operation allows the estimation of large displacement flow.

Our network, called PWC-Net, has been designed to make full use of these simple and well-established principles. It makes significant improvements in model size and accuracy over existing CNN models for optical flow (Fig. 1). PWC-Net is about 17 times smaller in size and 2 times faster in inferencing than FlowNet2. It is also easier to train than SpyNet and FlowNet2 and runs at about 35 frames per second (fps) on Sintel resolution (1024×436) images. It is the winning entry in the optical flow category of the first robust vision challenge.

- *D. Sun and J. Kautz are with NVIDIA, Westford, MA 01886. E-mail: {deqings, jkautz}@nvidia.com*
- *X. Yang and M.-Y. Liu are with NVIDIA, Santa Clara, CA 95050. E-mail: {xiaodongy, mingyul}@nvidia.com*



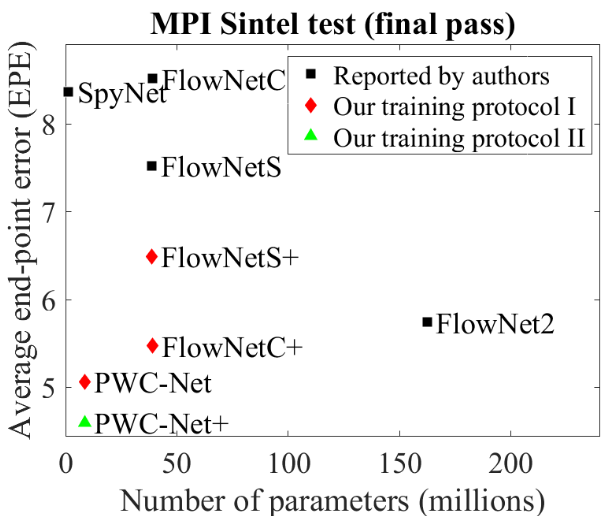


Fig. 1. Left: PWC-Net outperforms all published methods on the MPI Sintel final pass benchmark in both accuracy and running time. Right: compared with previous end-to-end CNN models for flow, PWC-Net reaches the best balance between accuracy and size. The comparisons among PWC-Net, FlowNetS+, and FlowNetC+ show the improvements brought by the network architectures; all have been trained using the same training protocols. The comparisons, FlowNetS vs. FlowNetS+, FlowNetC vs. FlowNetC+, and PWC-Net vs. PWC-Net+, show the improvements brought by training protocols. **Both models and training matter.**

However, it is imprecise or even misleading to conclude that the performance gains of PWC-Net come only from the new network architecture, because training matters as well. If trained improperly, a good model may perform poorly. CNNs were introduced in the late 80's [3], [18], but it took decades to figure out the details to train deep CNNs properly, such as dropout, ReLU units, batch normalization, and data augmentation [1]. For optical flow, Dosovitskiy *et al.* [8] report that FlowNetS outperforms FlowNetC. Ilg *et al.* [10] show that using more iterations and dataset scheduling results in improved performance for both FlowNetS and FlowNetC. In particular, FlowNetC performs better than FlowNetS with the new training procedure. For PWC-Net, we have used the same network architecture in the first version of our arXiv paper published in Sep. 2017, the second (CVPR) version in Nov. 2017, and the current one. The improvements over previous versions result solely from better training procedures.

In some sense, a straight forward comparison between PWC-Net and previous models is unfair, because the models have been trained differently. The potential of some models may be unfulfilled due to less than optimal training procedures. To fairly compare models, we retrain FlowNetC and FlowNetS, the sub-networks of FlwoNet2, using the same training protocol as PWC-Net. We observe significant performance improvements: the retrained FlowNetC is about 56% more accurate on Sintel final than the previously trained one, although still 8% less accurate than PWC-Net. A somewhat surprising result is that the retrained FlowNetC is about 5% more accurate on Sintel final compared to the published FlowNet2 model, which has a much larger capacity. The last comparison clearly shows that better training procedures may be more effective at improving the performance of a basic model than increasing the model size, because larger models are usually harder to train[1]. The results show a complicated interplay between models and training, which requires careful experimental designs to identify the sources of performance gains.

1. We cannot directly apply our training procedure to FlowNet2.

In this paper, we further improve the training procedures for PWC-Net. Specifically, adding KITTI and HD1K data during fine-tuning improves the average end-point error (EPE) of PWC-Net on Sintel final by 10% to 4.60, which is better than all published methods. Fixing an I/O bug, which incorrectly loaded 22% of the training data, leads to a $\sim$20% improvement on KITTI 2012 and 2015. At the time of writing, PWC-Net has the second lowest outlier percentage in non-occluded regions on KITTI 2015, surpassed only by a recent scene flow method that uses stereo input and semantic information [19].

To summarize, we make the following contributions:

- We present a compact and effective CNN model for optical flow based on well-established principles. It performs robustly across four major flow benchmarks and is the winning entry in the optical flow category of the robust vision challenge.
- We compare FlowNetS, FlowNetC and PWC-Net trained using the same training procedure. On Sintel final, the retrained FlowNetC is about 5% more accurate than the reported FlowNet2, which uses FlowNetC as a sub-network.
- We further improve the training procedures for Sintel and fix an I/O bug for KITTI, both resulting in significant performance gain for the same PWC-Net network architecture. The newly trained model parameters and training protocols will be available on https://github.com/NVlabs/PWC-Net.

2 Previous Work

Variational approach. Horn and Schunck [20] pioneer the variational approach to optical flow by coupling the brightness constancy and spatial smoothness assumptions using an energy function. Black and Anandan [21] introduce

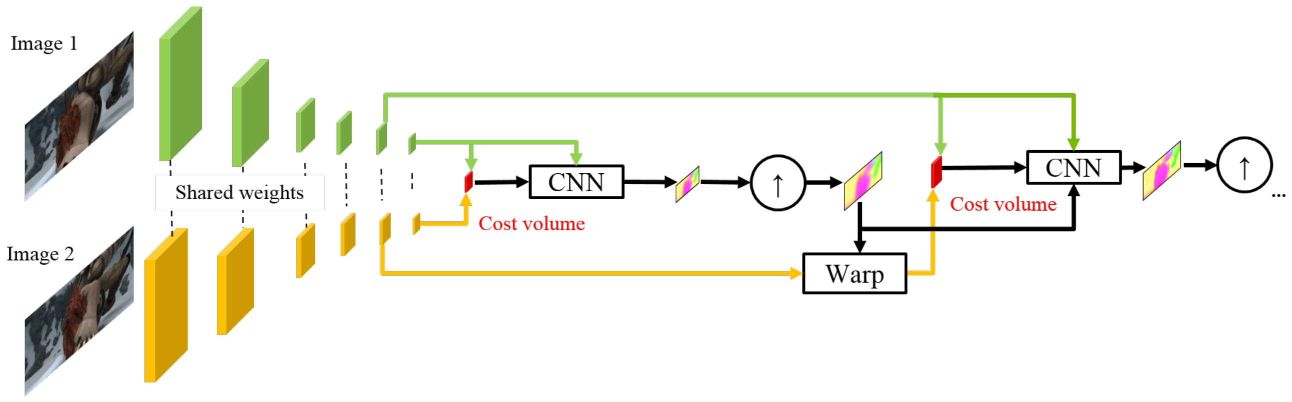


Fig. 2. **Network architecture of PWC-Net**. We only show the flow estimation modules at the top two levels. For the rest of the pyramidal levels, the flow estimation modules have the same structure as the second to top level.

a robust framework to deal with outliers, *i.e.*, brightness inconstancy and spatial discontinuities. As it is computationally impractical to perform a full search, a coarse-to-fine, warping-based approach is often adopted [22]. Brox *et al.* [23] theoretically justify the warping-based estimation process. The variational approach is the most popular framework for optical flow. However, it requires solving complex optimization problems and is computationally expensive for real-time applications.

Sun *et al.* [24] review the models, optimization, and implementation details for methods derived from Horn and Schunck. One surprising finding is that the original Horn and Schunck objective, when optimized using modern techniques and implementation details, performs competitively against contemporary state of the art. Thus, it is critical to separate the contributions from the objective and the optimization. Our work shows that it is also critical to separate the contributions from the CNN models and the training procedures.

One conundrum for the coarse-to-fine approach is small and fast moving objects that disappear at coarse levels. To address this issue, Brox and Malik [25] embed feature matching into the variational framework, which is further improved by follow-up methods [26], [27]. In particular, the EpicFlow method [28] can effectively interpolate sparse matches to dense optical flow and is widely used as a post-processing method [17], [29], [30], [31], [32], [33]. Zweig and Wolf [34] use CNNs for sparse-to-dense interpolation and obtain consistent improvement over EpicFlow.

Most top-performing methods use CNNs as a component in their system. For example, DCFlow [17], the best published method on MPI Sintel final pass so far, learns CNN features to construct a full cost volume and uses sophisticated post-processing techniques, including EpicFlow, to estimate the optical flow. The next-best method, FlowFieldsCNN [30], learns CNN features for sparse matching and densifies the matches by EpicFlow. The third-best method, MRFlow [35] uses a CNN to classify a scene into rigid and non-rigid regions and estimates the geometry and camera motion for rigid regions using a plane + parallax formulation. However, none of them are real-time or end-to-end trainable.

Early work on learning optical flow. There is a long history of learning optical flow before the deep learning era. Simoncelli and Adelson [36] study the data matching errors for optical flow. Freeman *et al.* [37] learn parameters of an MRF model for image motion using synthetic blob world examples. Roth and Black [38] study the spatial statistics of optical flow using sequences generated from depth maps. Sun *et al.* [39] learn a full model for optical flow, but the learning has been limited to a few training sequences [12]. Li and Huttenlocker [40] use stochastic optimization to tune the parameters for the Black and Anandan method [21], but the number of parameters learned is limited. Wulff and Black [41] learn PCA motion basis of optical flow estimated by GPUFlow [42] on real movies. Their method is fast but produces over-smoothed flow.

Recent work on learning optical flow. Inspired by the success of CNNs on high-level vision tasks [1], Dosovitskiy *et al.* [8] construct two CNN networks, FlowNetS and FlowNetC, for estimating optical flow based on the U-Net denoising autoencoder [9]. The networks are pre-trained on a large synthetic FlyingChairs dataset but can surprisingly capture the motion of fast moving objects on the Sintel dataset. The raw output of the network, however, contains large errors in smooth background regions and requires variational refinement [25]. Mayer *et al.* [43] apply the FlowNet architecture to disparity and scene flow estimation. Ilg *et al.* [10] stack several basic FlowNet models into a large one, *i.e.*, FlowNet2, which performs on par with state of the art on the Sintel benchmark. Ranjan and Black [11] develop a compact spatial pyramid network, called SpyNet. SpyNet achieves similar performance as the FlowNetC model on the Sintel benchmark, which is good but not state-of-the-art.

Another interesting line of research takes the unsupervised learning approach. Memisevic and Hinton [44] propose the gated restricted Boltzmann machine to learn image transformations in an unsupervised way. Long *et al.* [45] learn CNN models for optical flow by interpolating frames. Yu *et al.* [46] train models to minimize a loss term that combines a data constancy term with a spatial smoothness term. While inferior to supervised approaches on datasets with labeled training data, existing unsupervised methods can be used to (pre-)train CNN models on unlabeled data [47].

Cost volume. A cost volume stores the data matching costs for associating a pixel with its corresponding pixels at the next frame [14]. Its computation and processing are standard components for stereo matching, a special case of optical flow. Recent methods [8], [17], [31] investigate cost volume processing for optical flow. All build the full cost volume at a single scale, which is both computationally expensive and memory intensive. By contrast, our work shows that constructing a partial cost volume at multiple pyramid levels leads to both effective and efficient models.

Datasets. Unlike many other vision tasks, it is extremely difficult to obtain ground truth optical flow on real-world sequences. Early work on optical flow mainly relies on synthetic datasets [48], *e.g.*, the famous "Yosemite". Methods may over-fit to the synthetic data and do not perform well on real data [49]. Baker *et al.* [12] capture real sequences under both ambient and UV lights in a controlled lab environment to obtain ground truth, but the approach does not work for outdoor scenes. Liu *et al.* [49] use human annotations to obtain ground truth motion for natural video sequences, but the labeling process is time-consuming.

KITTI and Sintel are currently the most challenging and widely-used benchmarks for optical flow. The KITTI benchmark is targeted for autonomous driving applications and its semi-dense ground truth is collected using LIDAR [50]. The 2012 set only consists of static scenes. The 2015 set is extended to dynamic scenes via human annotations and more challenging to existing methods because of the large motion, severe illumination changes, and occlusions [51]. The Sintel benchmark [52] is created using the open source graphics movie "Sintel" with two passes, clean and final. The final pass contains strong atmospheric effects, motion blur, and camera noise, which cause severe problems to existing methods. All published, top-performing methods [17], [30], [35] rely heavily on traditional techniques. PWC-Net is the first fully end-to-end CNN model that outperforms all published methods on both the KITTI 2015 and Sintel final pass benchmarks.

CNN models for dense prediction tasks in vision. The denoising autoencoder [53] has been commonly used for dense prediction tasks in computer vision, especially with skip connections [9] between the encoder and decoder. Recent work [54], [55] shows that dilated convolution layers can better exploit contextual information and refine details for semantic segmentation. Here we use dilated convolutions to integrate contextual information for optical flow and obtain moderate performance improvement. The DenseNet architecture [7], [56] directly connects each layer to every other layer in a feedforward fashion and has been shown to be more accurate and easier to train than traditional CNN layers in image classification tasks. We test this idea for dense optical flow prediction.

3 Approach

We start from an overview of the network architecture of PWC-Net, as shown in Figure 2. PWC-Net first builds a feature pyramid from the two input images. At the top level of the pyramid, PWC-Net constructs a cost volume by comparing features of a pixel in the first image with corresponding features in the second image. As the top level is of small spatial resolution, we can construct the cost volume using a small search range. The cost volume and features of the first image are then fed to a CNN to estimate the flow at the top level. PWC-Net then upsamples and rescales the estimated flow to the next level. At the second to top level, PWC-Net warps features of the second image toward the first using the upsampled flow, and then constructs a cost volume using features of the first image and the warped features. As warping compensates the large motion, we can still use a small search range to construct the cost volume. The cost volume, features of the first image, and the upsampeld flow are fed to a CNN to estimate flow at the current level, which is then upsampled to the next (third) level. The process repeats until the desired level.

As PWC-Net has been designed using classical principles from optical flow, it is informative to compare the key components of PWC-Net with the traditional coarse-to-fine approaches [20], [21], [23], [24] in Figure 3. First, as raw images are variant to shadows and lighting changes [23], [24], we replace the fixed image pyramid with learnable feature pyramids. Second, we take the warping operation from the traditional approach as a layer in our network to estimate large motion. Third, as the cost volume is a more discriminative representation of the optical flow than raw images, our network has a layer to construct the cost volume, which is then processed by CNN layers to estimate the flow. The warping and cost volume layers have no learnable parameters and reduce the model size. Finally, a common practice by the traditional methods is to post-process the optical flow using contextual information, such as median filtering [57] and bilateral filtering [58]. Thus, PWC-Net uses a context network to exploit contextual information to refine the optical flow. Compared with energy minimization, the CNN models are computationally more efficient.

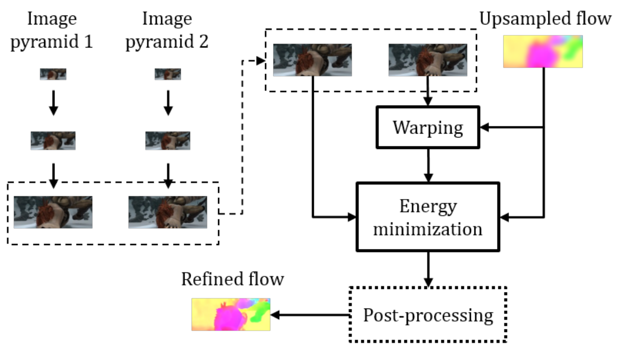
Next, we will explain the main ideas for each component, including pyramid feature extractor, optical flow estimator, and context networks. Please refer to the appendix for details of the networks.

Feature pyramid extractor. Given two input images $\mathbf{I}_1$ and $\mathbf{I}_2$, we generate L-level pyramids of feature representations, with the bottom (zeroth) level being the input images, *i.e.*, $\mathbf{c}_t^0 = \mathbf{I}_t$. To generate feature representation at the lth layer, $\mathbf{c}_t^l$, we use layers of convolutional filters to downsample the features at the $l-1$th pyramid level, $\mathbf{c}_t^{l-1}$, by a factor of 2. From the first to the sixth levels, the number of feature channels are respectively 16, 32, 64, 96, 128, and 192.

Warping layer. At the lth level, we first upsample by a factor of 2 and rescale the estimated flow from the $l+1$th level, $\mathbf{w}^{l+1}$, to the current level. We then warp features of the second image toward the first image using the upsampled flow:

$$\mathbf{c}_w^l(\mathbf{x}) = \mathbf{c}_2^l\big(\mathbf{x} + 2 \times \mathrm{up}_2(\mathbf{w}^{l+1})(\mathbf{x})\big), \tag{1}$$

where $\mathbf{x}$ is the pixel index and up_2 denote the $\times 2$ upsampling operator. We use bilinear interpolation to implement the warping operation and compute the gradients to the input CNN features and flow for backpropagation according to [10], [59]. For non-translational motion, warping can compensate for some geometric distortions and put image patches at the right scale. Note that there is no upsampled



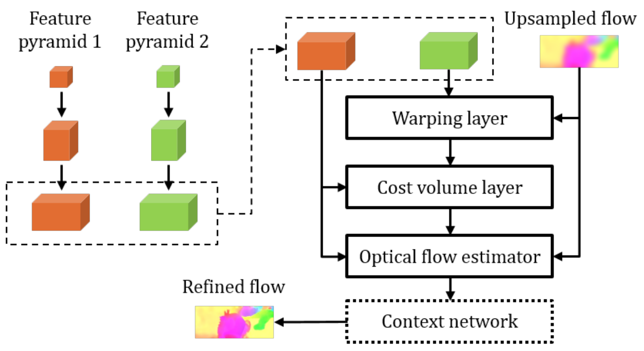


Fig. 3. **Traditional coarse-to-fine approach vs. PWC-Net.** Left: Image pyramid and refinement at one pyramid level by the energy minimization approach [20], [21], [23], [24]. Right: Feature pyramid and refinement at one pyramid level by PWC-Net. PWC-Net warps features of the second image using the upsampled flow, computes a cost volume, and process the cost volume using CNNs. Both post-processing and context network are optional in each system. The arrows indicate the direction of flow estimation and pyramids are constructed in the opposite direction. Please refer to the text for details about the network.

flow at the top pyramid level and the warped features are the same as features of the second image, *i.e.*, $\mathbf{c}_w^L = \mathbf{c}_2^L$.

Cost volume layer. Next, we use the features to construct a cost volume that stores the matching costs for associating a pixel with its corresponding pixels at the next frame [14]. We define the matching cost as the correlation [8], [17] between features of the first image and warped features of the second image:

$$\mathbf{cv}^l(\mathbf{x}_1, \mathbf{x}_2) = \frac{1}{N}\left(\mathbf{c}_1^l(\mathbf{x}_1)\right)^{\mathsf{T}} \mathbf{c}_w^l(\mathbf{x}_2), \tag{2}$$

where T is the transpose operator and N is the length of the column vector $\mathbf{c}_1^l(\mathbf{x}_1)$. For an L-level pyramid setting, we only need to compute a partial cost volume with a limited range of d pixels, *i.e.*, $|\mathbf{x}_1 - \mathbf{x}_2|_\infty \le d$. A one-pixel motion at the top level corresponds to 2^{L-1} pixels at the full resolution images. Thus we can set d to be small. The dimension of the 3D cost volume is $d^2 \times H^l \times W^l$, where H^l and W^l denote the height and width of the lth pyramid level, respectively.

Optical flow estimator. It is a multi-layer CNN. Its input are the cost volume, features of the first image, and upsampled optical flow and its output is the flow $\mathbf{w}^l$ at the lth level. The numbers of feature channels at each convolutional layers are respectively 128, 128, 96, 64, and 32, which are kept fixed at all pyramid levels. The estimators at different levels have their own parameters instead of sharing the same parameters. This estimation process is repeated until the desired level, l_0.

The estimator architecture can be enhanced with DenseNet connections [7]. The inputs to every convolutional layer are the output of and the input to its previous layer. DenseNet has more direct connections than traditional layers and leads to significant improvement in image classification. We test this idea for dense flow prediction.

Context network. Traditional flow methods often use contextual information to post-process the flow. Thus we employ a sub-network, called the context network, to effectively enlarge the receptive field size of each output unit at the desired pyramid level. It takes the estimated flow and features of the second last layer from the optical flow estimator and outputs a refined flow, $\hat{\mathbf{w}}_\Theta^{l_0}(\mathbf{x})$.

The context network is a feed-forward CNN and its design is based on dilated convolutions [55]. It consists of 7 convolutional layers. The spatial kernel for each convolutional layer is 3×3. These layers have different dilation constants. A convolutional layer with a dilation constant k means that an input unit to a filter in the layer are k-unit apart from the other input units to the filter in the layer, both in vertical and horizontal directions. Convolutional layers with large dilation constants enlarge the receptive field of each output unit without incurring a large computational burden. From bottom to top, the dilation constants are $1, 2, 4, 8, 16, 1$, and 1.

Training loss. Let Θ be the set of all the learnable parameters in our final network, which includes the feature pyramid extractor and the optical flow estimators at different pyramid levels (the warping and cost volume layers have no learnable parameters). Let $\mathbf{w}_\Theta^l$ denote the flow field at the lth pyramid level predicted by the network, and $\mathbf{w}_{\mathrm{GT}}^l$ the corresponding supervision signal. We use the same multi-scale training loss proposed in FlowNet [8]:

$$\mathcal{L}(\Theta) = \sum_{l=l_0}^{L} \alpha_l \sum_{\mathbf{x}} |\mathbf{w}_\Theta^l(\mathbf{x}) - \mathbf{w}_{\mathrm{GT}}^l(\mathbf{x})|_2 + \gamma|\Theta|_2^2, \tag{3}$$

where $|\cdot|_2$ computes the L2 norm of a vector and the second term regularizes parameters of the model. Note that if the context network is used at the l_0th level, $\mathbf{w}_\Theta^{l_0}$ will be replaced by the output of the context network, $\hat{\mathbf{w}}_\Theta^{l_0}(\mathbf{x})$. For fine-tuning, we use the following robust training loss:

$$\mathcal{L}(\Theta) = \sum_{l=l_0}^{L} \alpha_l \sum_{\mathbf{x}} \left(|\mathbf{w}_\Theta^l(\mathbf{x}) - \mathbf{w}_{\mathrm{GT}}^l(\mathbf{x})| + \epsilon\right)^q + \gamma|\Theta|_2^2 \tag{4}$$

where $|\cdot|$ denotes the L1 norm, $q < 1$ gives less penalty to outliers, and ϵ is a small constant.

4 Experimental Results

Implementation details. The weights in the training loss (3) are set to be $\alpha_6 = 0.32, \alpha_5 = 0.08, \alpha_4 = 0.02, \alpha_3 = 0.01$, and $\alpha_2 = 0.005$. The trade-off weight γ is set to be 0.0004. We scale the ground truth flow by 20 and downsample it to obtain the supervision signals at different levels. Note that we do not further scale the supervision signal at each level, the same as [8]. As a result, we need to scale the upsampled flow at each pyramid level for the warping layer.

For example, at the second level, we scale the upsampled flow from the third level by a factor of 5 $(= 20/4)$ before warping features of the second image. We use a 7-level pyramid ($L = 6$), consisting of 6 levels of CNN features and the input images as the bottom level. We set the desired level l_0 to be 2, *i.e.*, our model outputs a quarter resolution optical flow and uses bilinear interpolation to obtain the full-resolution optical flow. We use a search range of 4 pixels to compute the cost volume at each level.

We first train the models using the FlyingChairs dataset in Caffe [60] using the S_{long} learning rate schedule introduced in [10], *i.e.*, starting from 0.0001 and reducing the learning rate by half at 0.4M, 0.6M, 0.8M, and 1M iterations. The data augmentation scheme is the same as that in [10]. We crop 448×384 patches during data augmentation and use a batch size of 8. We then fine-tune the models on the FlyingThings3D dataset using the S_{fine} schedule [10] while excluding image pairs with extreme motion (magnitude larger than 1000 pixels). The cropped image size is 768×384 and the batch size is 4. Finally, we fine-tune the models using the Sintel and KITTI training sets and will explain the details below.

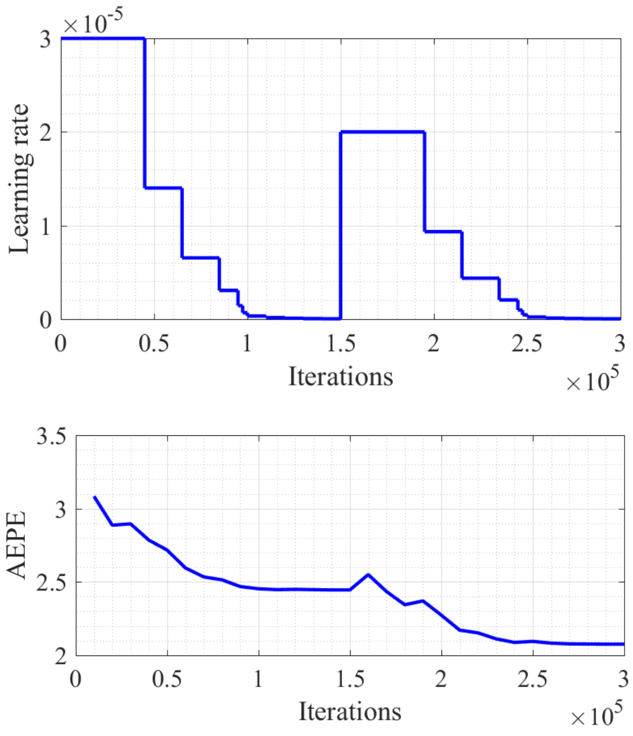


Fig. 4. Top: learning rate schedule for fine-tuning (the step values for the first 10^5 iterations were provided by Eddy Ilg). Bottom: average end-point error (EPE) on the final pass of the Sintel training set. We disrupt the learning rate for a better minimum, which has better accuracy in both the training and the test sets.

4.1 Main Results

4.1.1 *MPI Sintel.*

When fine-tuning on Sintel, we crop 768×384 image patches, add horizontal flip, and do not add additive Gaussian noise during data augmentation. The batch size is 4. We use the robust loss function in Eq. (4) with $\epsilon = 0.01$ and $q = 0.4$. We disrupt the learning rate, as shown in Fig. 4, which empirically improves both the training and test performance. We test two schemes of fine-tuning. The first one, PWC-Net-ft, uses the clean and final passes of the Sintel training data throughout the fine-tuning process. The second one, PWC-Net-ft-final, uses only the final pass for the second half of fine-tuning. We test the second scheme because DCFlow learns the features using only the final pass of the training data. Thus we test the performance of PWC-Net when the final pass of the training data is given more weight. We refer to the latter scheme as our **training protocol I**, which we will use later to train other models.

PWC-Net has lower average end-point error (EPE) than many recent methods on the final pass of the MPI-Sintel benchmark (Table 1). Further, PWC-Net is the fastest among all the top-performing methods (Fig. 1). We can further reduce the running time by dropping the DenseNet connections. The resulting PWC-Net-small model is about 5% less accurate but 40% faster than PWC-Net.

PWC-Net is less accurate than traditional approaches on the clean pass. Traditional methods often use image edges to refine motion boundaries, because the two are perfectly aligned in the clean pass. However, image edges in the final pass are corrupted by motion blur, atmospheric changes, and noise. Thus, the final pass is more realistic and challenging. The results on the final and clean sets suggest that PWC-Net may be better suited for real images, where the image edges are often corrupted.

PWC-Net has higher errors on the training set but lower errors on the test set than FlowNet2, suggesting that PWC-Net may have a more appropriate capacity for this task. Table 2 summarizes errors in different regions. PWC-Net performs relatively better in regions with large motion and away from the motion boundaries, probably because it has been trained using only data with large motion. Figure 5 shows the visual results of different variants of PWC-Net on the training and test sets of MPI Sintel. PWC-Net can recover sharp motion boundaries but may fail on small and rapidly moving objects, such as the left arm in “Market_5”.

4.1.2 *KITTI.*

When fine-tuning on KITTI, we crop 896×320 image patches and reduce the amount of rotation, zoom, and squeeze during data augmentation. The batch size is 4 too. The large patches can capture the large motion in the KITTI dataset. Since the ground truth is semi-dense, we upsample the predicted flow at the quarter resolution to compare with the scaled ground truth at the full resolution. We exclude the invalid pixels in computing the loss function.

The CVPR version of PWC-Net outperforms many recent two-frame optical flow methods on the 2015 set, as shown in Table 3. On the 2012 set, PWC-Net is inferior to SDF that assumes a rigidity constraint for the background. Although the rigidity assumption works well on the static scenes in the 2012 set, PWC-Net outperforms SDF in the 2015 set which mainly consists of dynamic scenes and is more challenging. The visual results in Fig. 6 qualitatively demonstrate the benefits of using the context network, DenseNet connections, and fine-tuning, respectively. In par-

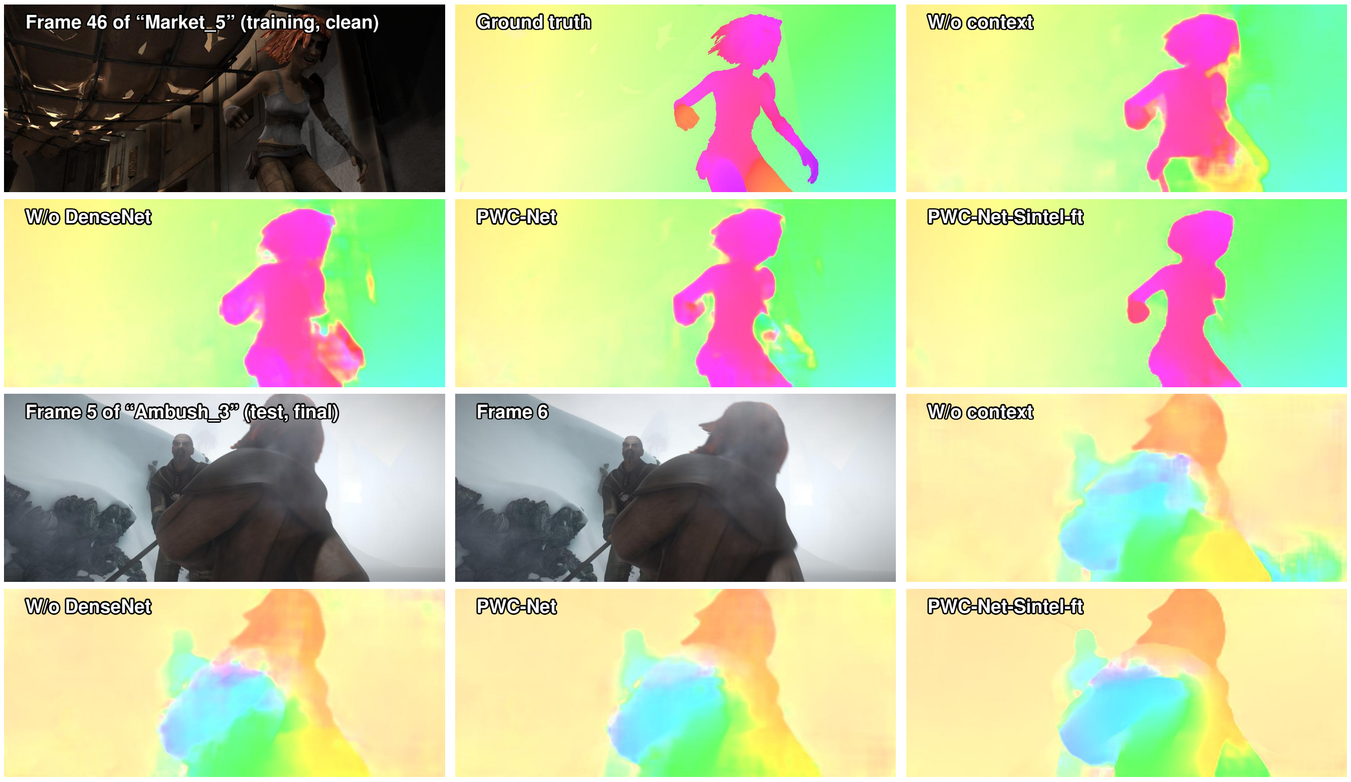


Fig. 5. Results on Sintel *training* and *test* sets. Context network, DenseNet connections, and fine-tuning all improve the results. Small and rapidly moving objects, *e.g.*, the left arm in "Market_5", are still challenging to the pyramid-based PWC-Net.

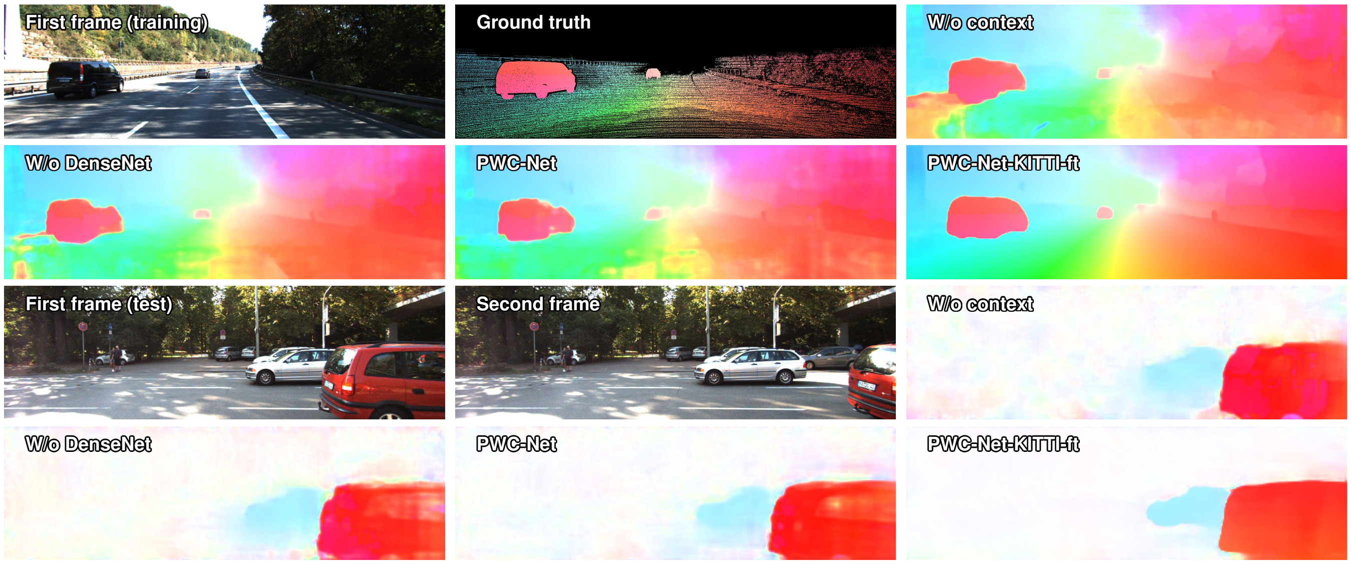


Fig. 6. Results on KITTI 2015 *training* and *test* sets. Fine-tuning fixes large regions of errors and recovers sharp motion boundaries.

ticular, fine-tuning fixes large regions of errors in the test set, demonstrating the benefit of learning when the training and test data share similar statistics.

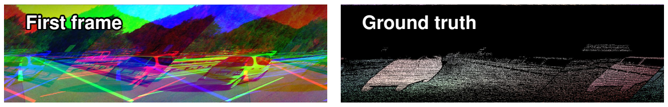


Fig. 7. Improperly read images and flow fields due to an I/O bug.

An I/O bug. We use the Caffe code [8] to make all the image pairs and flow fields into a single LMDB file for training. The code requires that all the input images are of the same resolution. The size of the first 156 sequences of KITTI 2015 is 375×1242, but the last 44 are of different resolutions, including 370×1224, 374×1238, and 376×1241. The Caffe code cannot read the last 44 sequences properly, as shown in Fig. 7. As a results, PWC-Net has been trained using only 156 "good" sequences and 44 "bad" ones. As a remedy, we crop all the sequences to the size of 370×1224, because there is no reliable way to resize the sparse ground truth. Re-training with the correct 200 sequences leads to about 20% improvement on the test set of KITTI 2012 (Fl-Noc 4.22% $\rightarrow$ 3.41%) and 2015 (Fl-all 9.60% $\rightarrow$ 7.90%). At the time of writing, PWC-Net is ranked second in non-occluded regions among all methods on KITTI 2015. It is surpassed by only one recent scene flow method that uses

TABLE 1
Average EPE results on MPI Sintel set. "-ft" means fine-tuning on the MPI Sintel *training* set and the numbers in the parenthesis are results on the data the methods have been fine-tuned on. ft-final gives more weight to the final pass during fine-tuning. FlowNetC2 has been trained using the same procedure as PWC-Net-ft-final.

Methods	Training		Test		Time
	Clean	Final	Clean	Final	(s)
PatchBatch [61]	-	-	5.79	6.78	50.0
EpicFlow [28]	-	-	4.12	6.29	15.0
CPM-flow [32]	-	-	3.56	5.96	4.30
FullFlow [31]	-	3.60	2.71	5.90	240
FlowFields [62]	-	-	3.75	5.81	28.0
MRFlow [35]	1.83	3.59	**2.53**	5.38	480
FlowFieldsCNN [30]	-	-	3.78	5.36	23.0
DCFlow [17]	-	-	3.54	5.12	8.60
SpyNet-ft [11]	(3.17)	(4.32)	6.64	8.36	0.16
FlowNet2 [10]	2.02	3.14	3.96	6.02	0.12
FlowNet2-ft [10]	(1.45)	(2.01)	4.16	5.74	0.12
LiteFlowNet-CVPR	(1.64)	(2.23)	4.86	6.09	0.09
LiteFlowNet-arXiv	(**1.35**)	(**1.78**)	4.54	5.38	0.09
FlowNetS+	(2.80)	(2.76)	6.49	6.54	**0.01**
FlowNetC+	2.31	2.34	5.04	5.47	0.05
PWC-Net-small	2.83	4.08	-	-	0.02
PWC-Net-small-ft	(2.27)	(2.45)	5.05	5.32	0.02
PWC-Net	2.55	3.93	-	-	0.03
PWC-Net-ft	(1.70)	(2.21)	3.86	5.13	0.03
PWC-Net-ft-final	(2.02)	(2.08)	4.39	5.04	0.03
PWC-Net_ROB	(1.81)	(2.29)	3.90	4.90	0.03
PWC-Net+	(1.71)	(2.34)	3.45	**4.60**	0.03

TABLE 2
Detailed results on the Sintel benchmark for different regions, velocities (s), and distances from motion boundaries (d).

Final	matched	unmatched	d_{0-10}	d_{10-60}	d_{60-140}	s_{0-10}	s_{10-40}	s_{40+}
PWC-Net	**2.44**	**27.08**	**4.68**	**2.08**	**1.52**	**0.90**	**2.99**	**31.28**
FlowNet2	2.75	30.11	4.82	2.56	1.74	0.96	3.23	35.54
SpyNet	4.51	39.69	6.69	4.37	3.29	1.40	5.53	49.71
Clean								
PWC-Net	**1.45**	**23.47**	3.83	**1.31**	**0.56**	0.70	2.19	**23.56**
FlowNet2	1.56	25.40	**3.27**	1.46	0.86	**0.60**	**1.89**	27.35
SpyNet	3.01	36.19	5.50	3.12	1.72	0.83	3.34	43.44

stereo input and semantic information [19] (Fl-all scores: 5.07% vs 4.69%) and more accurate than other scene flow methods, such as another recent one that also uses semantic information [64]. Note that scene flow methods can use the estimated depth and the camera motion to predict the flow of out-of-boundary pixels and thus tend to have better accuracy in all regions.

4.1.3 Robust Vision Challenge[2]

PWC-Net_ROB is the winning entry in the optical flow competition of the robust vision challenge, which requires applying a method using the same parameter setting to four benchmarks: Sintel [52], KITTI 2015 [51], HD1K [65], and Middlebury [12]. To participate the challenge, we fine-tune the model using training data from Sintel, KITTI 2015, and HD1K and name it as PWC-Net_ROB. We do not use the Middlebury training data because the provided eight image pairs are too small and of low resolution compared to other datasets. We use a batch size of 6, with 2 image pairs from Sintel, KITTI, and HD1K respectively. The cropping size is 768×320 for Sintel and KITTI 2015. For HD1K, we first crop 1536×640 patches and then downsample the cropped images and flow fields to 768×320. For the mixed datasets, we use more iterations and learning rate disruptions, as shown in Fig. 8.

2. http://www.robustvision.net

TABLE 3
Results on the KITTI dataset. "-ft" means fine-tuning on the KITTI *training* set and the numbers in the parenthesis are results on the data the methods have been fine-tuned on.

Methods	KITTI 2012			KITTI 2015		
	AEPE	AEPE	Fl-Noc	AEPE	Fl-all	Fl-all
	train	*test*	*test*	*train*	*train*	*test*
EpicFlow [28]	-	3.8	7.88%	-	-	26.29 %
FullFlow [31]	-	-	-	-	-	23.37 %
CPM-flow [32]	-	3.2	5.79%	-	-	22.40 %
PatchBatch [61]	-	3.3	5.29%	-	-	21.07%
FlowFields [62]	-	-	-	-	-	19.80%
MRFlow [35]	-	-	-	-	14.09 %	12.19 %
DCFlow [17]	-	-	-	-	15.09 %	14.83 %
SDF [29]	-	2.3	3.80%	-	-	11.01 %
MirrorFlow [63]	-	2.6	4.38%	-	9.93%	10.29%
SpyNet-ft [11]	(4.13)	4.7	12.31%	-	-	35.07%
FlowNet2 [10]	4.09	-	-	10.06	30.37%	-
FlowNet2-ft [10]	(1.28)	1.8	4.82%	(2.30)	(8.61%)	10.41 %
LiteFlowNet-CVPR	(1.26)	1.7	-	(2.16)	(8.16%)	10.24 %
LiteFlowNet-arXiv	(**1.05**)	1.6	**3.27%**	(1.62)	(**5.58%**)	9.38 %
PWC-Net	4.14	-	-	10.35	33.67%	-
PWC-Net-ft-CVPR	(1.45)	1.7	4.22%	(2.16)	(9.80%)	9.60%
PWC-Net-ft	(1.08)	**1.5**	3.41%	(**1.45**)	(7.59%)	**7.90%**

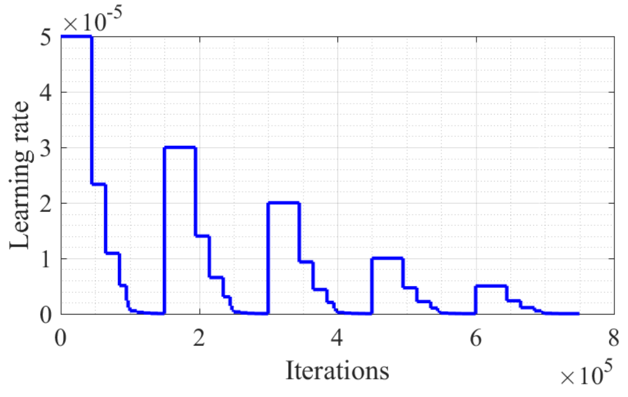


Fig. 8. Learning rate schedule for fine-tuning using data from Sintel, KITTI, and HD1K. For this mixed dataset, we use more iterations and learning rate disruptions than the learning rate schedule in Fig. 4.

Using mixed datasets increases the test error on KITTI 2015 (F-all 9.60% $\rightarrow$ 11.63%) but reduces the test error on MPI Sintel final (AEPE 5.04% $\rightarrow$ 4.9). There is a larger mismatch between the training and test data of Sintel than those of KITTI 2015. Thus, using more diverse datasets reduces the over-fitting errors on Sintel. We further use a batch size of 4, with 2 image pairs from Sintel, 1 from KITTI, and 1 from HD1K respectively, which is our **training procotol II**. It results in a further performance gain, *i.e.*, PWC-Net+ in Table 2.

The Middlebury images are of lower resolution and we upsample them so that the larger of the width and height of the upsampled image is around 1000 pixels. PWC-Net_ROB has similar performance as the Classic+NL method (avg. training EPE 0.24 vs. 0.22; avg. test EPE 0.33 vs 0.32).

PWC-Net_ROB is ranked first on the HD1K benchmark [65], which consists of real-world images corrupted by rain, glare, and windshield wipers *etc.*. The 2560×1080 res-

TABLE 4
Ablation experiments. Unless explicitly stated, the models have been trained on the FlyingChairs dataset.

	Chairs	Sintel Clean	Sintel Final	KITTI 2012 AEPE	KITTI 2012 Fl-all	KITTI 2015 AEPE	KITTI 2015 Fl-all
Full model	2.00	3.33	4.59	5.14	28.67%	13.20	41.79%
Feature ↑	**1.92**	**3.03**	**4.17**	**4.57**	**26.73%**	**11.64**	**39.80%**
Feature ↓	2.18	3.36	4.56	5.75	30.79%	14.05	44.92%
Image	2.95	4.42	5.58	7.28	31.25%	16.29	45.13%

(a) Larger-capacity **feature pyramid extractor** has better performance. Learning features leads to significantly better results than fixed image pyramids.

Max. Disp.	Chairs	Sintel Clean	Sintel Final	KITTI 2012 AEPE	KITTI 2012 Fl-all	KITTI 2015 AEPE	KITTI 2015 Fl-all
0	2.13	3.66	5.09	5.25	29.82%	13.85	43.52%
2	2.09	**3.30**	**4.50**	5.26	**25.99%**	13.67	**38.99%**
Full model (4)	2.00	3.33	4.59	5.14	28.67%	13.20	41.79%
6	**1.97**	3.31	4.60	**4.96**	27.05%	**12.97**	40.94%

(b) **Cost volume.** Removing the cost volume (0) results in moderate performance loss. PWC-Net can handle large motion using a small search range to compute the cost volume.

	Trained on FlyingChairs			Fine-tuned on FlyingThings		
	Chairs	Clean	Final	Chairs	Clean	Final
5-level	2.13	3.28	4.52	2.62	2.98	4.29
6-level	**1.95**	**2.96**	**4.32**	**2.28**	**2.50**	3.97
Full model (7)	2.00	3.33	4.59	2.30	2.55	**3.93**

(c) **More feature pyramid levels** help after fine-tuning on FlyingThings.

	Chairs	Sintel Clean	Sintel Final	KITTI 2012 AEPE	KITTI 2012 Fl-all	KITTI 2015 AEPE	KITTI 2015 Fl-all
Full model	2.00	3.33	4.59	5.14	28.67%	13.20	41.79%
Estimator ↑	**1.92**	**3.09**	**4.50**	**4.64**	**25.34%**	**12.25**	**39.18%**
Estimator ↓	2.01	3.37	4.58	4.82	26.35%	12.83	40.53%

(d) Larger-capacity **optical flow estimator** has better performance.

	Trained on FlyingChairs			Fine-tuned on FlyingThings		
	Chairs	Clean	Final	Chairs	Clean	Final
Full model	**2.00**	3.33	4.59	**2.34**	**2.60**	**3.95**
No DenseNet	2.06	**3.09**	**4.37**	2.48	2.83	4.08
No Context	2.23	3.47	4.74	2.55	2.75	4.13

(e) **Context network** consistently helps; **DenseNet** helps after fine-tuning on FlyingThings.

	Chairs	Sintel Clean	Sintel Final	KITTI 2012 AEPE	KITTI 2012 Fl-all	KITTI 2015 AEPE	KITTI 2015 Fl-all
1st run	2.00	3.33	4.59	5.14	28.67%	13.20	41.79%
2nd run	2.00	**3.23**	**4.36**	**4.70**	**25.52%**	**12.57**	**39.06%**
3rd run	2.00	3.33	4.65	4.81	27.12%	13.10	40.84%

(f) Independent runs with different initializations lead to minor performance differences.

	Chairs	Sintel Clean	Sintel Final	KITTI 2012 AEPE	KITTI 2012 Fl-all	KITTI 2015 AEPE	KITTI 2015 Fl-all
Full model	**2.00**	**3.33**	**4.59**	**5.14**	**28.67%**	**13.20**	**41.79%**
No warping	2.17	3.79	5.30	5.80	32.73%	13.74	44.87%

(g) **Warping layer** is a critical component for the performance.

	Chairs	Sintel Clean	Sintel Final	KITTI 2012 AEPE	KITTI 2012 Fl-all	KITTI 2015 AEPE	KITTI 2015 Fl-all
Full model	2.00	3.33	4.59	5.14	28.67%	13.20	41.79%
Residual	**1.96**	**3.14**	**4.43**	**4.87**	**27.74%**	**12.58**	**41.16%**

(h) **Residual connections** in the optical flow estimator are helpful.

olution images causes out-of-memory issue on an NVIDIA Pascal TitanX GPU with 12GB memory and requires an NVIDIA Volta 100 GPU with 16GB memory. Figure 9 shows some visual results on Middlebury and HD1K test set. Despite minor artifacts, PWC-Net_ROB performs robustly across these benchmarks using the same set of parameters.

4.2 Ablation Experiments

Feature pyramid extractor. PWC-Net uses a two-layer CNN to extract features at each pyramid level. Table 5a summarizes the results of two variants that use one layer (↓) and three layers (↑) respectively. A larger-capacity feature pyramid extractor leads to consistently better results on both the training and validation datasets. Replacing the feature pyramids with image pyramids results in about 40% loss in accuracy, confirming the benefits of learning features.

To further understand the effect of the pyramids, we test feature pyramids with different levels, as shown in Table 5c. Using 5-level pyramids leads to consistently worse results. Using 6-level pyramids has better performance than the default 7-level pyramids when trained on FlyingChairs, but the two have close performance after fine-tuning using FlyingThings3D. One possible reason is that the cropping size for FlyingChairs (448 × 384) is too small for the 7-level pyramids. The size of the top level is 7 × 6, too small for a search range of 4 pixels. By contrast, the cropping size for FlyingThings3D (768 × 384) is better suited for the 7-level-pyramids.

Optical flow estimator. PWC-Net uses a five-layer CNN in the optical flow estimator at each level. Table 5d shows the results by two variants that use four layer (↓) and seven layers (↑) respectively. A larger-capacity optical flow estimator leads to better performance. However, we observe in our experiments that a deeper optical flow estimator might get stuck at poor local minima, which can be detected by checking the validation errors after a few thousand iterations and fixed by running from a different random initialization.

Removing the context network results in larger errors on both the training and validation sets (Table 5e). Removing the DenseNet connections results in higher training error but lower validation errors when the model is trained on FlyingChairs. However, after the model is fine-tuned on FlyingThings3D, DenseNet leads to lower errors.

We also test a residual version of the optical flow estimator, which estimates a flow increment and adds it to the initial flow to obtain the refined flow. As shown in Table 5h, this residual version slightly improves the performance.

Cost volume. We test the search range to compute the cost volume, shown in Table 5b. Removing the cost volume results in consistent worse results. A larger range leads to lower training error. However, all three settings have similar performance on Sintel, because a range of 2 at every level can already deal with a motion up to 200 pixels at the input resolution. A larger range has lower EPE on KITTI, likely because the images from the KITTI dataset have larger displacements than those from Sintel. A smaller range, however, seems to force the network to ignore pixels with extremely large motion and focus more on small-motion pixels, thereby achieving lower Fl-all scores.

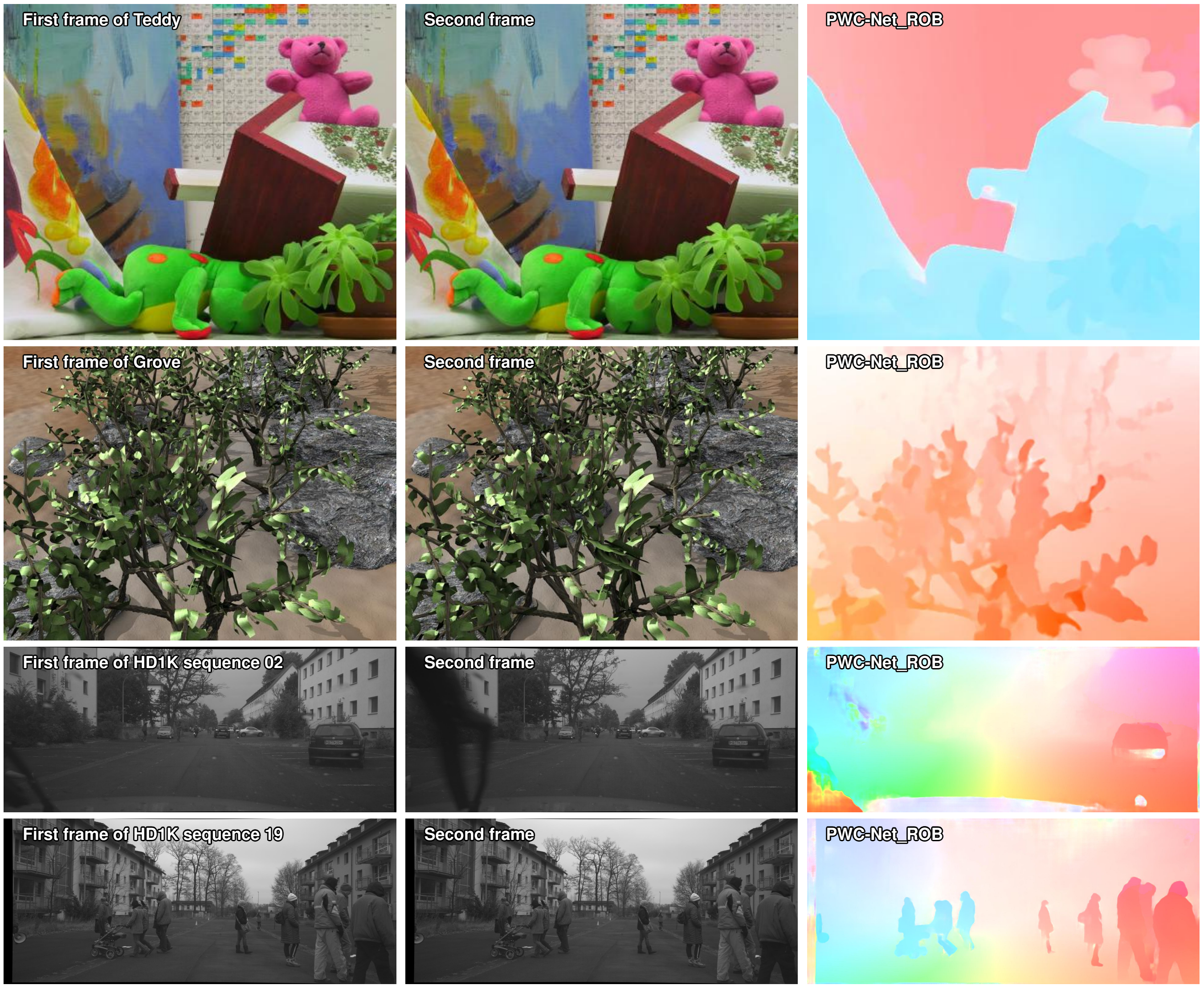


Fig. 9. Results on Middlebury and HD1K test sets. PWC-Net_ROB has not been trained using the training data of Middlebury but performs reasonably well on the test set. It cannot recover the fine motion details of the twigs in Grove though. PWC-Net_ROB has reasonable results in the regions occluded by the windshield wipers in sequence 02 of the HD1K test set.

Warping. Warping allows for estimating a small optical flow (increment) at each pyramid level to deal with a large optical flow. Removing the warping layers results in a significant loss of accuracy (Table 5g). Without the warping layer, PWC-Net still produces reasonable results, because the default search range of 4 to compute the cost volume is large enough to capture the motion of most sequences at the low-resolution pyramid levels.

Independent Runs. To test the robustness to the initializations, we train PWC-Net with different runs. These independent runs have almost the same training error but some minor differences in performance on the validation sets, as shown in Table 5f.

Dataset scheduling. We also train PWC-Net using different dataset scheduling schemes, as shown in Table 5. Sequentially training on FlyingChairs, FlyingThings3D, and Sintel gradually improves the performance, consistent with the observations in [10]. Directly training using the test data leads to good "over-fitting" results, but the trained model does not perform as well on other datasets.

Model size and running time. Table 6 summarizes the model size for different CNN models. PWC-Net has about 17 times fewer parameters than FlowNet2. PWC-Net-small further reduces this by an additional 2 times via dropping DenseNet connections and is more suitable for memory-limited applications.

The timings have been obtained on the same desktop with an NVIDIA Pascal TitanX GPU. For more precise timing, we exclude the reading and writing time when benchmarking the forward and backward inference time. PWC-Net is about 2 times faster in forward inference and at least 3 times faster in training than FlowNet2.

TABLE 5
Training dataset schedule leads to better local minima. () indicates results on the dataset the method has been trained on.

Data	Chairs	Sintel (AEPE)		KITTI 2012		KITTI 2015	
	AEPE	Clean	Final	AEPE	Fl-all	AEPE	Fl-all
Chairs	**(2.00)**	3.33	4.59	5.14	28.67%	13.20	41.79%
Chairs-Things	2.30	2.55	3.93	4.14	21.38%	10.35	33.67%
Chairs-Things-Sintel	2.56	**(1.70)**	**(2.21)**	**2.94**	**12.70%**	**8.15**	**24.35%**
Sintel	3.69	(1.86)	(2.31)	3.68	16.65%	10.52	30.49%

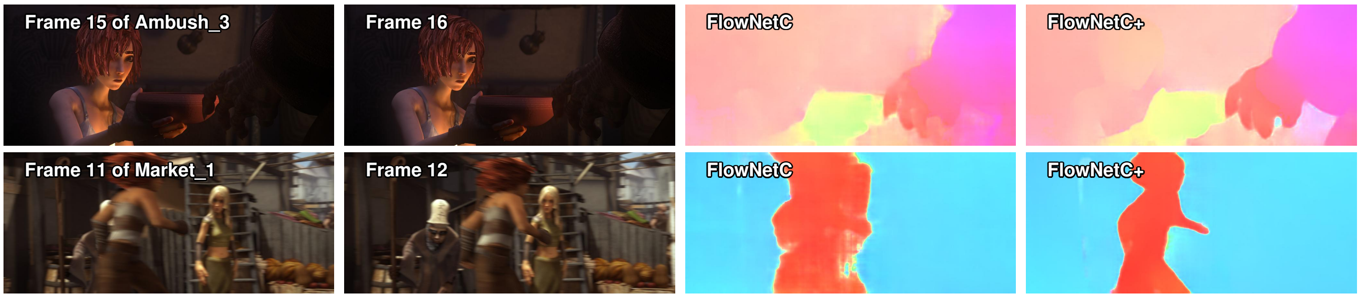


Fig. 10. **Training procedure matters**. FlowNetC and FlowNetC+ use the same network architecture but have been trained differently. FlowNetC+ has been trained using our procedure and generates results with finer details and fewer artifacts than the previously trained FlowNetC.

TABLE 6
Model size and running time. PWC-Net-small drops DenseNet connections. For training, the lower bound of 14 days for FlowNet2 is obtained by 6(FlowNetC) + 2×4 (FlowNetS). The inference time is for 1024×448 resolution images.

Methods	FlowNetS	FlowNetC	FlowNet2	SpyNet	PWC-Net	PWC-Net-small
#parameters (M)	38.67	39.17	162.49	1.2	8.75	4.08
Parameter Ratio	23.80%	24.11%	100%	0.74%	5.38%	2.51%
Memory (MB)	154.5	156.4	638.5	9.7	41.1	22.9
Memory Ratio	24.20%	24.49%	100%	1.52%	6.44%	3.59%
Training (days)	4	6	>14	-	4.8	4.1
Forward (ms)	11.40	21.69	84.80	-	28.56	20.76
Backward (ms)	16.71	48.67	78.96	-	44.37	28.44

4.3 Training Matters

We used the same architecture in the first [66] and second (CVPR) versions of our arXiv paper but observed an about 10% improvement on the final pass of the Sintel test set. The performance improvement results solely from changes in the training procedures, including performing horizontal flips, not adding additive Gaussian noise, and disrupting the learning rate. One questions arises: how do other models perform using the same training procedure as PWC-Net?

To better understand the effects of models and training and fairly compare with existing methods, we re-train the FlowNetS and FlowNetC models using exactly the same training procedure as PWC-Net, including the robust training loss function. We name the retrained model as FlowNetS+ and FlowNetC+ respectively and evaluate them using the test set of Sintel, as summarized in Table 1. Figure 10 shows the visual results FlowNetC trained using different training protocols. The results by FlowNetC+ have fewer artifacts and are more piece-wise smooth than the reviously trained FlowNetC. As shown in Table 1, FlowNetC+ is about 8% less accurate on Sintel final and 3 times larger in model size than PWC-Net , which demonstrates the benefit of the new network architecture under the same training procedure.

To our surprise, FlowNetC+ is about 5% more accurate than the published FlowNet2 model on the final pass, because FlowNet2 uses FlowNetC as a sub-network. We should note that this is not a fair comparison for FlowNet2, because we are unable to apply the same training protocol to the FlowNet2 model, which requires sequential training of several sub-networks. It is expected that a more careful, customized training schemes would improve the performance of FlowNet2.

It is often assumed or taken for granted that the results published by authors represent the best possible performance of a method. However, our results show that we should carefully evaluate published methods to "identify the source of empirical gains" [67]. When we observe improvements over previous models, it is critical to analyze whether the gain results from the model or the training procedure. It would be informative to evaluate models trained in the same way or compare training procedures using the same model. To enable fair comparisons and further innovations, we will make our training protocols available.

5 Comparison With Closely-related Work

As the field is changing rapidly, it is informative to do a detailed discussion of the closely-related work. Both FlowNet2 [10] and SpyNet [11] have been designed using principles from stereo and optical flow. However, the architecture of PWC-Net has significant differences.

SpyNet uses image pyramids while PWC-Net learns feature pyramids. FlowNet2 uses three-level feature pyramids in the first module of its whole network, *i.e.*, FlowNetC. By contrast, PWC-Net uses much deeper feature pyramids. As analyzed in the ablation study, using deeper feature pyramids usually leads to better performance. Both SpyNet and FlowNet2 warp the input images, while PWC-Net warps the features, which enables the information to propagate throughout the whole feature pyramids.

SpyNet feeds CNNs with images, while PWC-Net feeds a cost volume. As the cost volume is a more discriminative representation of the search space for optical flow, the learning task for CNNs becomes easier. FlowNet2/FlowNetC constructs the cost volume at a single resolution with a large search range. However, using features at a fixed resolution may not be effective at resolving the well-known "aperture problem" [20], [68], [69], [70]. By contrast, PWC-Net constructs multi-resolution cost volume and reduces the computation using a small search range.

Regarding performance, PWC-Net outperforms SpyNet by a significant margin. Additionally, SpyNet has been trained sequentially, while PWC-Net can be trained end-to-end from scratch. FlowNet2 achieves impressive performance by stacking several basic models into a large-capacity model. The much smaller PWC-Net obtains similar or better performance by embedding classical principles into the network architecture. It would be interesting to use PWC-Net as a building block to design large networks.

Two recent papers also incorporate domain knowledge of flow into the CNN architectures. LiteFlowNet [71] uses similar ideas as PWC-Net, including feature pyramids, warping features, and constructing a cost volume with a limited search range at multiple resolutions. LiteFlowNet furthers incorporates a flow regularization layer to deal with outliers using a feature-driven local convolutions. However, LiteFlowNet requires sequential (stage-wise) training. The CVPR final version of LiteFlowNet (March. 2018) is about 8% less accurate on Sintel final than the first arXiv version of PWC-Net [66] published in Sep. 2017 (avg. EPE 6.09 vs. 5.63). In an updated arXiv version [72] published in May 2018, LiteFlowNet uses similar data augmentation schemes as the CVPR final version of PWC-Net, *e.g.*, not adding Gaussian noise, horizontal flipping (image mirroring), and reducing the spatial data augmentation for KITTI. With these changes, LiteFlowNet reports performance close to PWC-Net on Sintel final (avg. EPE: 5.33 vs 5.04) and KITTI 2015 (F-all: 9.38% vs. 9.60%). This further confirms the importance of training in obtaining top performance.

Another paper, TVNet [73], subsumes a specific optical flow solver, the TV-L1 method [74], and is initialized by unfolding its optimization iterations as neural layers. TVNet is used to learn rich and task-specific patterns and obtains excellent performance on activity classification. The readers are urged to read these papers to better understand the similarities and differences.

TABLE 7
Comparison of network architectures.

Principles	FlowNetC	FlowNet2	SpyNet	PWC-Net
Pyramid	3-level	3-level	Image	6-level
Warping	-	Image	Image	Feature
Cost volume	single-level large range	single-level large range	-	multi-level small range

6 Conclusions

We have developed a compact but effective CNN model for optical flow estimation using simple and well-established principles: pyramidal processing, warping, and cost volume processing. Combining deep learning with domain knowledge not only reduces the model size but also improves the performance. PWC-Net is about 17 times smaller in size, 2 times faster in inference, easier to train, and 11% more accurate on Sintel final than the recent FlowNet2 model. It performs robustly across four different benchmarks using the same set of parameters and is the winning entry in the optical flow competition of the robust vision challenge.

We have also shown that the performance gains of PWC-Net result from both the new model architecture and the training procedures. Retrained using our procedures, FlowNetC is even 5% more accurate on Sintel final than the published FlowNet2, which uses FlowNetC as a sub-network. We have further improved the training procedures, which increase the accuracy of PWC-Net on Sintel by 10% and on KITTI 2012 and 2015 by 20%. The results show the complicated interplay between models and training and call for careful experimental designs to identify the sources of empirical gains. To enable comparison and further innovations, we will make the retrained models and training protocols available on https://github.com/NVlabs/PWC-Net.

Acknowledgements We would like to thank Zhile Ren and Jinwei Gu for porting the Caffe code to PyTorch, Eddy Ilg for clarifying details about the FlowNet2 paper, Ming-Hsuan Yang for helpful suggestions, Michael Pellauer for proofreading, github users for clarifying questions, and the anonymous reviewers at ICCV'17 and CVPR'18 for constructive comments.

References

[1] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems (NIPS)*, 2012. 1, 2, 3
[2] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein *et al.*, "Imagenet large scale visual recognition challenge," *International Journal of Computer Vision (IJCV)*, vol. 115, no. 3, pp. 211–252, 2015. 1
[3] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel, "Backpropagation applied to handwritten zip code recognition," *Neural computation*, 1989. 1, 2
[4] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014. 1
[5] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015. 1
[6] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. 1
[7] G. Huang, Z. Liu, K. Q. Weinberger, and L. van der Maaten, "Densely connected convolutional networks," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017. 1, 4, 5
[8] A. Dosovitskiy, P. Fischery, E. Ilg, C. Hazirbas, V. Golkov, P. van der Smagt, D. Cremers, T. Brox *et al.*, "FlowNet: Learning optical flow with convolutional networks," in *IEEE International Conference on Computer Vision (ICCV)*, 2015. 1, 2, 3, 4, 5, 7
[9] O. Ronneberger, P. Fischer, and T. Brox, "U-Net: Convolutional networks for biomedical image segmentation," in *International Conference on Medical Image Computing and Computer Assisted Intervention (MICCAI)*, 2015. 1, 3, 4
[10] E. Ilg, N. Mayer, T. Saikia, M. Keuper, A. Dosovitskiy, and T. Brox, "FlowNet 2.0: Evolution of optical flow estimation with deep networks," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017. 1, 2, 3, 4, 6, 8, 10, 11
[11] A. Ranjan and M. J. Black, "Optical flow estimation using a spatial pyramid network," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017. 1, 3, 8, 11
[12] S. Baker, D. Scharstein, J. P. Lewis, S. Roth, M. J. Black, and R. Szeliski, "A database and evaluation methodology for optical flow," *International Journal of Computer Vision (IJCV)*, 2011. 1, 3, 4, 8
[13] J. Weber and J. Malik, "Robust computation of optical flow in a multi-scale differential framework," *International Journal of Computer Vision (IJCV)*, vol. 14, no. 1, pp. 67–81, 1995. 1
[14] A. Hosni, C. Rhemann, M. Bleyer, C. Rother, and M. Gelautz, "Fast cost-volume filtering for visual correspondence and beyond," *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 2013. 1, 3, 5
[15] D. Scharstein and R. Szeliski, "A taxonomy and evaluation of dense two-frame stereo correspondence algorithms," *International Journal of Computer Vision (IJCV)*, 2002. 1
[16] J. Zbontar and Y. LeCun, "Stereo matching by training a convolutional neural network to compare image patches," *Journal of Machine Learning Research (JMLR)*, 2016. 1
[17] J. Xu, R. Ranftl, and V. Koltun, "Accurate optical flow via direct cost volume processing," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017. 1, 3, 4, 5, 8
[18] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998. 2
[19] A. Behl, O. H. Jafari, S. K. Mustikovela, H. A. Alhaija, C. Rother, and A. Geiger, "Bounding boxes, segmentations and object coordinates: How important is recognition for 3D scene flow estimation in autonomous driving scenarios?" in *IEEE International Conference on Computer Vision (ICCV)*, 2017. 2, 8

[20] B. Horn and B. Schunck, "Determining optical flow," *Artificial Intelligence*, 1981. 2, 4, 5, 11

[21] M. J. Black and P. Anandan, "The robust estimation of multiple motions: Parametric and piecewise-smooth flow fields," *Computer Vision and Image Understanding (CVIU)*, 1996. 2, 3, 4, 5

[22] A. Bruhn, J. Weickert, and C. Schnörr, "Lucas/Kanade meets Horn/Schunck: combining local and global optic flow methods," *International Journal of Computer Vision (IJCV)*, 2005. 3

[23] T. Brox, A. Bruhn, N. Papenberg, and J. Weickert, "High accuracy optical flow estimation based on a theory for warping," in *European Conference on Computer Vision (ECCV)*, 2004. 3, 4, 5

[24] D. Sun, S. Roth, and M. J. Black, "A quantitative analysis of current practices in optical flow estimation and the principles behind them," *International Journal of Computer Vision (IJCV)*, 2014. 3, 4, 5

[25] T. Brox and J. Malik, "Large displacement optical flow: Descriptor matching in variational motion estimation," *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 2011. 3

[26] P. Weinzaepfel, J. Revaud, Z. Harchaoui, and C. Schmid, "DeepFlow: Large displacement optical flow with deep matching," in *IEEE International Conference on Computer Vision (ICCV)*, 2013. 3

[27] L. Xu, J. Jia, and Y. Matsushita, "Motion detail preserving optical flow estimation," *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 2012. 3

[28] J. Revaud, P. Weinzaepfel, Z. Harchaoui, and C. Schmid, "EpicFlow: Edge-preserving interpolation of correspondences for optical flow," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015. 3, 8

[29] M. Bai, W. Luo, K. Kundu, and R. Urtasun, "Exploiting semantic information and deep matching for optical flow," in *European Conference on Computer Vision (ECCV)*, 2016. 3, 8

[30] C. Bailer, K. Varanasi, and D. Stricker, "CNN-based patch matching for optical flow with thresholded hinge embedding loss," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017. 3, 4, 8

[31] Q. Chen and V. Koltun, "Full flow: Optical flow estimation by global optimization over regular grids," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. 3, 4, 8

[32] Y. Hu, R. Song, and Y. Li, "Efficient coarse-to-fine patchmatch for large displacement optical flow," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. 3, 8

[33] Y. Yang and S. Soatto, "S2f: Slow-to-fast interpolator flow," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017. 3

[34] S. Zweig and L. Wolf, "Interponet, a brain inspired neural network for optical flow dense interpolation," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017. 3

[35] J. Wulff, L. Sevilla-Lara, and M. J. Black, "Optical flow in mostly rigid scenes," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017. 3, 4, 8

[36] E. P. Simoncelli, E. H. Adelson, and D. J. Heeger, "Probability distributions of optical flow," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 1991. 3

[37] W. T. Freeman, E. C. Pasztor, and O. T. Carmichael, "Learning low-level vision," *International Journal of Computer Vision (IJCV)*, 2000. 3

[38] S. Roth and M. J. Black, "On the spatial statistics of optical flow," *International Journal of Computer Vision (IJCV)*, 2007. 3

[39] D. Sun, S. Roth, J. P. Lewis, and M. J. Black, "Learning optical flow," in *European Conference on Computer Vision (ECCV)*, 2008. 3

[40] Y. Li and D. P. Huttenlocher, "Learning for optical flow using stochastic optimization," in *European Conference on Computer Vision (ECCV)*, 2008. 3

[41] J. Wulff and M. J. Black, "Efficient sparse-to-dense optical flow estimation using a learned basis and layers," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015, pp. 120–130. 3

[42] M. Werlberger, W. Trobin, T. Pock, A. Wedel, D. Cremers, and H. Bischof, "Anisotropic Huber-L1 optical flow," in *British Machine Vision Conference (BMVC)*, 2009. 3

[43] N. Mayer, E. Ilg, P. Häusser, P. Fischer, D. Cremers, A. Dosovitskiy, and T. Brox, "A large dataset to train convolutional networks for disparity, optical flow, and scene flow estimation," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. 3

[44] R. Memisevic and G. Hinton, "Unsupervised learning of image transformations," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2007. 3

[45] G. Long, L. Kneip, J. M. Alvarez, H. Li, X. Zhang, and Q. Yu, "Learning image matching by simply watching video," in *European Conference on Computer Vision (ECCV)*, 2016. 3

[46] J. J. Yu, A. W. Harley, and K. G. Derpanis, "Back to basics: Unsupervised learning of optical flow via brightness constancy and motion smoothness," in *CoRR*, 2016. 3

[47] W.-S. Lai, J.-B. Huang, and M.-H. Yang, "Semi-supervised learning for optical flow with generative adversarial networks," in *Advances in Neural Information Processing Systems (NIPS)*, 2017. 3

[48] J. Barron, D. Fleet, and S. Beauchemin, "Performance of optical flow techniques," *International Journal of Computer Vision (IJCV)*, 1994. 4

[49] C. Liu, W. T. Freeman, E. H. Adelson, and Y. Weiss, "Human-assisted motion annotation," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2008. 4

[50] A. Geiger, P. Lenz, and R. Urtasun, "Are we ready for autonomous driving? The KITTI vision benchmark suite," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012. 4

[51] M. Menze and A. Geiger, "Object scene flow for autonomous vehicles," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015. 4, 8

[52] D. J. Butler, J. Wulff, G. B. Stanley, and M. J. Black, "A naturalistic open source movie for optical flow evaluation," in *European Conference on Computer Vision (ECCV)*, 2012. 4, 8

[53] P. Vincent, H. Larochelle, Y. Bengio, and P.-A. Manzagol, "Extracting and composing robust features with denoising autoencoders," in *International Conference on Machine Learning (ICML)*, 2008. 4

[54] L. C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille, "DeepLab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected CRFs," *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 2017. 4

[55] F. Yu and V. Koltun, "Multi-scale context aggregation by dilated convolutions," in *International Conference on Learning Representations (ICLR)*, 2016. 4, 5

[56] S. Jégou, M. Drozdzal, D. Vazquez, A. Romero, and Y. Bengio, "The one hundred layers tiramisu: Fully convolutional densenets for semantic segmentation," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshop*, 2017. 4

[57] A. Wedel, T. Pock, C. Zach, D. Cremers, and H. Bischof, "An improved algorithm for TV-L1 optical flow," in *Dagstuhl Motion Workshop*, 2008. 4

[58] J. Xiao, H. Cheng, H. Sawhney, C. Rao, and M. Isnardi, "Bilateral filtering-based optical flow estimation with occlusion detection," in *European Conference on Computer Vision (ECCV)*, 2006. 4

[59] M. Jaderberg, K. Simonyan, A. Zisserman *et al.*, "Spatial transformer networks," in *Advances in Neural Information Processing Systems (NIPS)*, 2015. 4

[60] 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 *ACM Multimedia*, 2014. 6

[61] D. Gadot and L. Wolf, "PatchBatch: A batch augmented loss for optical flow," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. 8

[62] C. Bailer, B. Taetz, and D. Stricker, "Flow fields: Dense correspondence fields for highly accurate large displacement optical flow estimation," in *IEEE International Conference on Computer Vision (ICCV)*, 2015. 8

[63] J. Hur and S. Roth, "MirrorFlow: Exploiting symmetries in joint optical flow and occlusion estimation," in *IEEE International Conference on Computer Vision (ICCV)*, Oct 2017. 8

[64] Z. Ren, D. Sun, J. Kautz, and E. Sudderth, "Cascaded scene flow prediction using semantic segmentation," in *3DV*, 2017. 8

[65] D. Kondermann, R. Nair, K. Honauer, K. Krispin, J. Andrulis, A. Brock, B. Gussefeld, M. Rahimimoghaddam, S. Hofmann, C. Brenner *et al.*, "The hci benchmark suite: Stereo and flow ground truth with uncertainties for urban autonomous driving," in *CVPR Workshops*, 2016, pp. 19–28. 8

[66] D. Sun, X. Yang, M.-Y. Liu, and J. Kautz, "PWC-Net: CNNs for optical flow using pyramid, warping, and cost volume," *arXiv preprint arXiv:1709.02371*, 2017. 11, 12

[67] Z. C. Lipton and J. Steinhardt, "Troubling trends in machine learning scholarship," *arXiv preprint arXiv:1807.03341*, 2018. 11

[68] E. H. Adelson and J. A. Movshon, "Phenomenal coherence of moving visual patterns," *Nature*, vol. 300, no. 5892, p. 523, 1982. 11

[69] B. Lucas and T. Kanade, "An iterative image registration technique with an application to stereo vision," 1981, pp. 674–679. 11

[70] Y. Weiss, E. P. Simoncelli, and E. H. Adelson, "Motion illusions as optimal percepts," *Nature neuroscience*, vol. 5, no. 6, p. 598, 2002. 11
[71] T.-W. Hui, X. Tang, and C. Change Loy, "Liteflownet: A lightweight convolutional neural network for optical flow estimation," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018. 11
[72] T. Hui, X. Tang, and C. C. Loy, "Liteflownet: A lightweight convolutional neural network for optical flow estimation," *arXiv preprint arXiv:1805.07036*, 2018. 12
[73] L. Fan, W. Huang, C. Gan, S. Ermon, B. Gong, and J. Huang, "End-to-end learning of motion representation for video understanding," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018. 12
[74] C. Zach, T. Pock, and H. Bischof, "A duality based approach for realtime tv-l 1 optical flow," in *German Conference on Pattern Recognition (DAGM)*, 2007. 12

APPENDIX

In this appendix, we proivde more details about PWC-Net. Figure 11 shows the architecture for the 7-level feature pyramid extractor network used in our experiment. Note that the bottom level consists of the original input images. Figure 12 shows the optical flow estimator network at pyramid level 2. The optical flow estimator networks at other levels have the same structure except for the top level, which does not have the upsampled optical flow and directly computes cost volume using features of the first and second images. Figure 13 shows the context network that is adopted only at pyramid level 2.

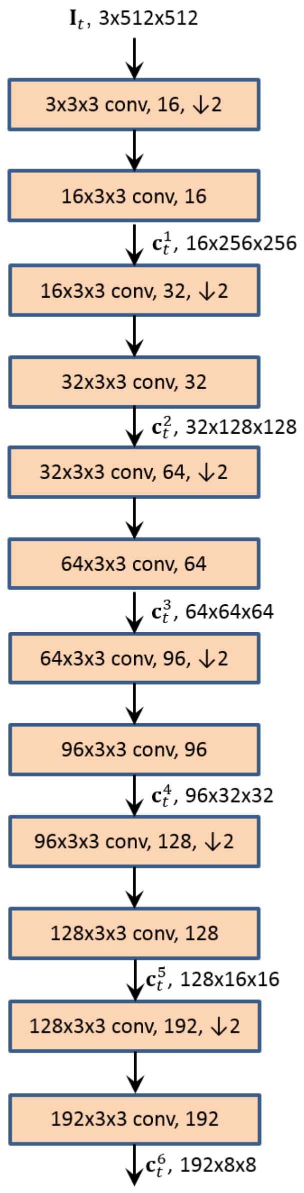


Fig. 11. The feature pyramid extractor network. The first image ($t = 1$) and the second image ($t = 2$) are encoded using the same Siamese network. Each convolution is followed by a leaky ReLU unit. The convolutional layer and the $\times 2$ downsampling layer at each level is implemented using a single convolutional layer with a stride of 2. $\mathbf{c}_t^l$ denotes extracted features of image t at level l;

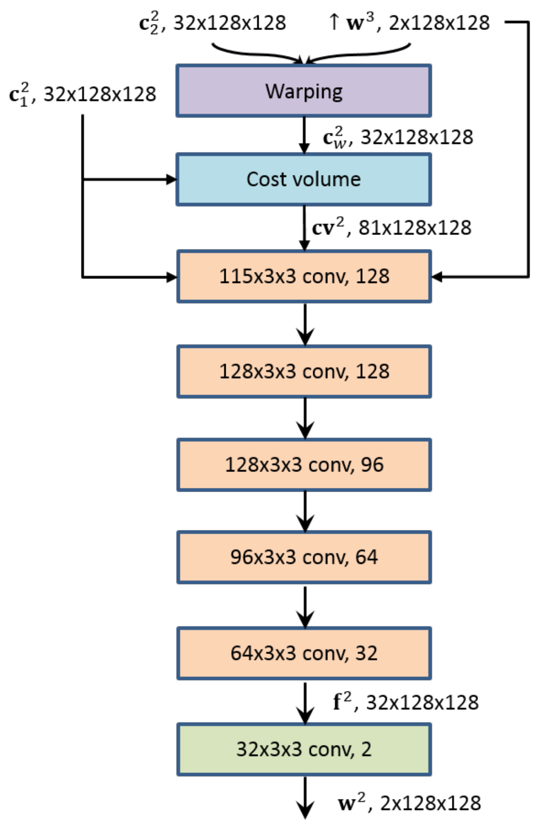

$\mathbf{c}_2^2$, 32x128x128
$\uparrow \mathbf{w}^3$, 2x128x128
$\mathbf{c}_1^2$, 32x128x128
Warping
$\mathbf{c}_w^2$, 32x128x128
Cost volume
$\mathbf{cv}^2$, 81x128x128
115x3x3 conv, 128
128x3x3 conv, 128
128x3x3 conv, 96
96x3x3 conv, 64
64x3x3 conv, 32
$\mathbf{f}^2$, 32x128x128
32x3x3 conv, 2
$\mathbf{w}^2$, 2x128x128

Fig. 12. The optical flow estimator network at pyramid level 2. Each convolutional layer is followed by a leaky ReLU unit except the last (light green) one that outputs the optical flow.

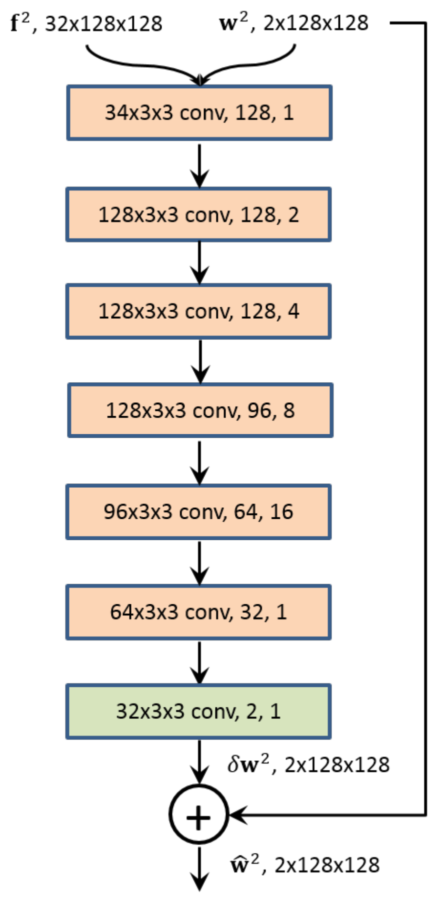


Fig. 13. The context network at pyramid level 2. Each convolutional layer is followed by a leaky ReLU unit except the last (light green) one that outputs the optical flow. The last number in each convolutional layer denotes the dilation constant.