Temporal Interpolation as an Unsupervised Pretraining Task for Optical Flow Estimation

Jonas Wulff[1,2] Michael J. Black[1]

[1] Max-Planck Institute for Intelligent Systems, Tübingen, Germany
[2] MIT CSAIL, Cambridge, MA, USA
wulff@mit.edu, black@tuebingen.mpg.de

Abstract. The difficulty of annotating training data is a major obstacle to using CNNs for low-level tasks in video. Synthetic data often does not generalize to real videos, while unsupervised methods require heuristic losses. Proxy tasks can overcome these issues, and start by training a network for a task for which annotation is easier or which can be trained unsupervised. The trained network is then fine-tuned for the original task using small amounts of ground truth data. Here, we investigate frame interpolation as a proxy task for optical flow. Using real movies, we train a CNN unsupervised for temporal interpolation. Such a network implicitly estimates motion, but cannot handle untextured regions. By fine-tuning on small amounts of ground truth flow, the network can learn to fill in homogeneous regions and compute full optical flow fields. Using this unsupervised pre-training, our network outperforms similar architectures that were trained supervised using synthetic optical flow.

1 Introduction

In recent years, the successes of deep convolutional neural networks (CNNs) have led to a large number of breakthroughs in most fields of computer vision. The two factors that made this possible are a widespread adoption of massively parallel processors in the form of GPUs and large amounts of available annotated training data. To learn good and sufficiently general representations of visual features, CNNs require tens of thousands to several hundred million [28] examples of the visual content they are supposed to process, annotated with labels teaching the CNNs the desired output for a given visual stimulus.

For some tasks such as image classification or object detection it is possible to generate massive amounts of data by paying human workers to manually annotate images. For example, writing a description of an image into a textbox or dragging a rectangle around an object are tasks that are easy to understand and relatively quick to perform. For other tasks, especially those related to video, obtaining ground truth is not as easy. For example, manual annotation of dense optical flow, motion segmentation, or tracking of objects requires not just the annotation of a huge number of instances (in the first two cases one would ideally want one annotation per pixel), but an annotator would also have to step back and forth in time to ensure temporal consistency [17]. Since this makes the task

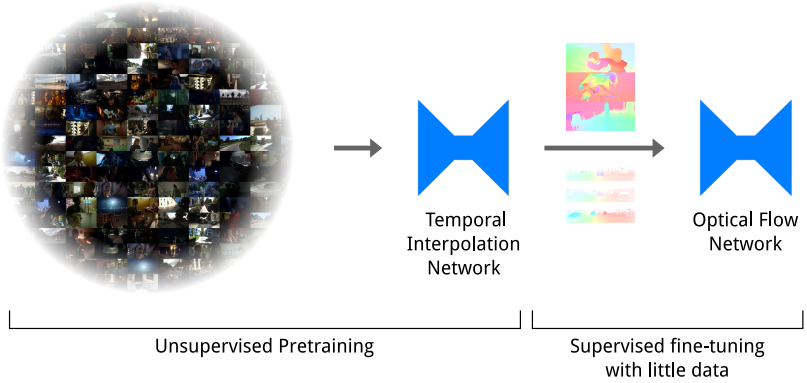

Fig. 1: Overview of our method. Using large amounts of unlabelled data we train a temporal interpolation network without explicit supervision. We then fine-tune the network for the task of optical flow using small amounts of ground truth data, outperforming similar architectures that were trained supervised.

both tedious and error-prone, manually annotated data is rarely used for low-level video analysis.

An alternative is to use synthetic data, for example from animated movies [4], videogames [26], or generated procedurally [19]. The problem here is that synthetic data always lives in a world different from our own. Even if the low-level image statistics are a good match to those of the real world, it is an open question whether realistic effects such as physics or human behavior can be learned from synthetic data.

A different approach to address the issue of small data is to use *proxy tasks*. The idea is to train a network using data for which labels are easy to acquire, or for which unsupervised or self-supervised learning is possible. Training on such data forces the network to learn a latent representation, and if the proxy task is appropriately chosen, this representation transfers to the actual task at hand. The trained network, or parts thereof, can then be fine-tuned to the final task using limited amounts of annotated data, making use of the structure learned from the proxy task. An example proxy task for visual reasoning about images is colorization, since solving the colorization problem requires the network to learn about the semantics of the world [16]. Another example is context prediction [5], in which the proxy task is to predict the spatial arrangement of sub-patches of objects, which helps in the final task of object detection.

What, then, would be a good proxy task for video analysis? One core problem in the analysis of video is to compute temporal correspondences between frames. Once correspondences are established, it is possible to reason about the temporal evolution of the scene, track objects, classify actions, and reconstruct the geometry of the world. Recent work by Long *et al.* [18] proposed a way to learn about correspondences without supervision. They first train a network to interpolate between two frames. For each point in the interpolated frame,

they then backpropagate the derivatives through the network, computing which pixels in the input images most strongly influence this point. This, in turn, establishes correspondences between the two maxima of these derivatives in the input images. Their work, however, had two main shortcomings. First, using a complete backpropagation pass to compute each correspondence is computationally expensive. Second, especially in unstructured or blurry regions of the frame, the derivatives are not necessarily well located, since a good (in the photometric sense) output pixel can be sampled from a number of wrongly corresponding sites in the input images; frame interpolation does not need to get the flow right to produce a result with low photometric error. This corresponds to the classical aperture problem in optical flow, in which the flow is not locally constrained, but context is needed to resolve ambiguities. Consequently, so far, the interpolation task has not served as an effective proxy for learning flow.

In this work, we address these shortcomings and show that, treated properly, the interpolation task can, indeed, support the learning of optical flow. To this end, we treat training for optical flow as a two-stage problem. In the first stage, we train a network to estimate the center frame from four adjacent, equally spaced frames. This forces the network to learn to establish correspondences in visually distinct areas. Unlike previous work, which used only limited datasets of a few tens of thousands frames such as KITTI-RAW [8], we use a little under one million samples from a diverse set of datasets incorporating both driving scenarios and several movies and TV series. This trains the network to better cope with effects like large displacements and motion and focus blur. Thanks to this varied and large body of training data, our network outperforms specialized frame interpolation methods despite not being tailored to this task.

In a second stage, we fine-tune the network using a small amount of ground-truth optical flow data from the training sets of KITTI [8] and Sintel [4]. This has three advantages. First, after fine-tuning, the network outputs optical flow directly, which makes it much more efficient than [18]. Second, this fine-tuning forces the network to group untextured regions and to consider the context when estimating the motion; as mentioned above, this can usually not be learned from photometric errors alone. Third, compared to fully unsupervised optical flow algorithms [1, 21], during training our method does not employ prior assumptions such as spatial smoothness, but is purely data-driven.

Our resulting network is fast and yields optical flow results with superior accuracy to the comparable networks of FlowNetS [6] and SpyNet [24] which were trained using large amounts of labeled, synthetic optical flow data [24]. This demonstrates that (a) when computing optical flow, it is important to use real data for training, and (b) that temporal interpolation is a suitable proxy task to learn from to make learning from such data feasible.

2 Previous work

CNNs for Optical Flow. In the past years, end-to-end training has had considerable success in many tasks of computer vision, including optical flow. The first

paper demonstrating end-to-end optical flow was FlowNet [6], which used an architecture similar to ours, but trained on large amounts of synthetic ground truth optical flow. In follow-up work [12], the authors propose a cascade of hourglass networks, each warping the images closer towards each other. Furthermore, they significantly extend their training dataset (which is still synthetic). This leads to high performance at the cost of a complicated training procedure.

Taking a different direction, SpyNet [24] combines deep learning with a spatial pyramid. Similar to classical optical flow methods, each pyramid level computes the flow residual for the flow on the next coarser scale, and successively warps the input frame using the new, refined flow. This allows the authors to use a very simple network architecture, which in turns leads to high computational efficiency. The training, however, is still done using the same annotated training set as [6]. The recently proposed PWC-Net [31] uses ideas of both, and computes the flow in a multiscale fashion using a cost volume on each scale, followed by a trained flow refinement step.

A different approach is to not train a network for full end-to-end optical flow estimation, but to use trained networks inside a larger pipeline. Most commonly, these approaches use a network to estimate the similarity between two patches [11, 34], effectively replacing the data cost in a variational flow method by a CNN. The resulting data costs can be combined with spatial inference, for example belief-propagation [11], or a cost volume optimization [34]. A network trained to denoise data can also be used as the proximal operator in a variational framework [20]. In the classical optical flow formulation, this would correspond to a network that learns to regularize.

All these approaches, however, require large amounts of annotated data. For real sequences, such training data is either hard to obtain for general scenes [14], or limited to specific domains such as driving scenarios [8]. Synthetic benchmarks [4, 19, 26], on the other hand, often lack realism, and it is unclear how well their training data generalizes to the real world.

Hence, several works have investigated unsupervised training for optical flow estimation. A common approach is to let the network estimate a flow field, warp the input images towards each other using this flow field, and measure the similarity of the images under a photometric loss. Since warping can be formulated as a differentiable function [13], the photometric loss can be back-propagated, forcing the network to learn better optical flow. In [35], the authors combine the photometric loss with a robust spatial loss on the estimated flow, similar to robust regularization in variational optical flow methods [29]. However, while their training is unsupervised, they do not demonstrate cross-dataset generalization, but train for Flying Chairs [6] and KITTI [8] using matching training sets and separately tuned hyper-parameters. In [25], the authors use the same approach, but show that a network that is pre-trained using the same dataset as in [6] can be fine-tuned to different output scenarios. Similarly, USCNN [1] uses only a single training set. Here, the authors do not use end-to-end training, but instead use a photometric loss to train a network to estimate the residual flow on different pyramid layers, similar to [24]. The recently proposed UnFlow [21]

uses a loss based on the CENSUS transform and computes the flow in both forward and backward direction. This allows the authors to integrate occlusion reasoning into the loss; using an architecture based on FlowNet2, they achieve state-of-the-art results on driving scenarios. All these methods require manually chosen heuristics as part of the loss, such as spatial smoothness or forward-backward consistency-based occlusion reasoning. Therefore, they do not fully exploit the fact that CNNs allow us to overcome such heuristics and to purely "let the data speak". In contrast, our method does not use any explicitly defined heuristics, but uses an unsupervised interpolation task and a small number of ground truth flow fields to learn about motion.

Several approaches use geometrical reasoning to self-supervise the training process. In TransFlow [2], the authors train two networks, a first, shallow one estimating a global homography between two input frames, and a second, deep network estimating the residual flow after warping with this homography. Since they use the homography to model the ego-motion, they focus on driving scenarios, and do not test on more generic optical flow sequences. In [9], a network is trained to estimate depth from a single image, and the photometric error according to the warping in a known stereo setup induced by the depth is penalized. Similarly [36] trains a network to estimate depth from a single image by learning to warp, but use videos instead of stereo. SfM-Net [7] learns to reduce a photometric loss by estimating the 3D structure of the world, the motion of the observer, and the segmentation into moving and static regions, which is enough to explain most of the motion of the world. However, as most other methods, it is only tested on the restricted automotive scenario.

Simliar to self-supervision using geometric losses, Generative Adversarial Networks [10] have been used to learn the structure of optical flow fields. In [15], the GAN is trained to distinguish between the warping errors caused by ground truth and wrongly estimated optical flow. Only the discriminator uses annotated data; once trained, it provides a loss for unsupervised training of the flow itself.

Frame interpolation. Instead of warping one input frame to another, it is also possible to train networks to interpolate and extrapolate images by showing them unlabeled videos at training time. A hybrid network is used in [27], where a shared contractive network is combined with two expanding networks to estimate optical flow and to predict the next frame, respectively. Similar to us, they hypothesize that temporal frame generation and optical flow share some internal representations; however, unlike us they train the optical flow network completely with labeled data, and do not test on the challenging Sintel test set. Similarly, [32] trains a network to anticipate the values of intermediate feature maps in the future. However, they are not interested in the motion itself, but in the future higher-level scene properties such as objects and actions. Niklaus *et al.* [23] propose a video interpolation network, where the motion is encoded in an estimated convolutional kernel; however, the quality of this implicit motion is never tested. As mentioned above, the work most similar to ours is [18], where a CNN is trained to interpolate between frames and subsequently used to compute correspondences between images. Unlike our work, however, they

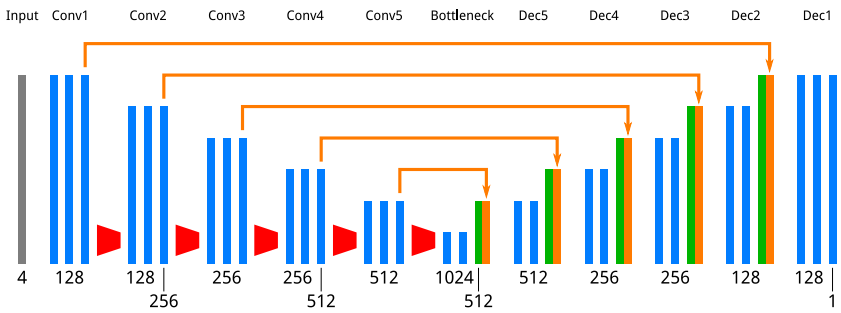

Fig. 2: Architecture of our network. The outputs of 3×3 convolutional layers are shown in blue, the output of 2×2 max-pooling operations in red, $2\times$ transposed convolutions in green, and side-channel concatenation in orange. Not shown are leaky ReLUs after each layer except the last and batch normalization layers. The numbers indicate the number of channels of each feature map.

Fig. 3: Our network predicts the center frame (green) from four neighboring, equally spaced frames (red). To ensure equal spacing of the input frames, the unmarked frames (second from the left and right) are not taken into account.

require expensive backpropagation steps to establish the correspondences; we show that using a small amount of training data can teach the network to translate between its internal motion representation and optical flow, resulting in significantly improved performance.

3 A frame interpolation network

The core hypothesis of our work is that in order to properly perform temporal interpolation, it is necessary to learn about the motion of the world. Temporal interpolation, however, is a task that does not require explicit supervision; with proper selection of input and output frames, every video sequence can serve as a supervisory signal. Therefore, we start by training a network for the task of interpolation in an unsupervised manner. Given four frames (as shown in Fig. 3 in red), the task of our network is to interpolate the center frame, shown in green in Fig. 3. Empirically, we found that using four input frames resulted in approximately 13% lower errors (2.04 px EPE) on the optical flow estimation task than using two frames (2.31 px EPE). We believe that the reasons for this are that with more than two frames (a) it is easier to reason about higher order temporal effects (*ie.* non-zero acceleration), and (b) it enables the network to reason about occlusions, which requires at least three frames [30]. We hence use four input frames for both the interpolation and the optical flow estimation task.

We use grayscale images as input and output, since we found the final optical flow to have lower errors with grayscale than with color images.

Network architecture. Similar to [18], we use a standard hourglass architecture with side-channels, as shown in Fig. 2. This simple architecture is nevertheless surprisingly effective in many different applications, such as optical flow computation [6] and unsupervised depth estimation [36]. Our network consists of five convolutional blocks (`Conv1` to `Conv5` in Fig. 2), each of which contains three 3×3 convolutional layers followed by batch normalization layers and leaky ReLUs. Between the blocks, we use max-pooling to reduce the resolution by a factor of two. Within each block, all layers except the last output the same number of feature maps. `Conv5` is followed by a bottleneck block consisting of two convolutional layers and a transposed convolution. The output is then successively upscaled using a series of decoder blocks (`Dec5` to `Dec1`). Each consists of two convolutional layers and (except for `Dec1`) a transposed convolution which doubles the resolution, again interleaved with leaky ReLUs and batch normalization layers.

To preserve high frequency information, we use side channels as in [6], shown in orange in Fig. 2. The output of the transposed convolutions are concatenated with the appropriate outputs from the convolutional layers. The last convolutional layer of `Dec1` directly outputs the monochrome output image and is not followed by a nonlinearity. Table 1 summarizes the number of inputs and outputs of each convolutional block.

Table 1: Number of input and output channels per layer block.

	Conv1	Conv2	Conv3	Conv4	Conv5	Bottleneck	Dec5	Dec4	Dec3	Dec2	Dec1
Input	4	128	128	256	256	512	1024	1024	512	512	256
Output	128	128	256	256	512	1024	1024	512	512	256	1

Training data. Unsupervised training would in theory allow us to train a network with infinite amounts of data. Yet, most previous works only use restricted datasets, such as KITTI-RAW. Instead, in this work we aim to compile a large, varied dataset, containing both cinematic sequences from several movies of different genres as well as more restricted but very common sequences from driving scenarios. As movies, we use *Victoria* and *Birdman*. The former was filmed in a true single take, and the later was filmed and edited in such a way that cuts are imperceptible. In addition, we use long takes from the movies *Atonement*, *Children of Men*, *Baby Driver*, and the TV series *True Detective*. Shot boundaries would destroy the temporal consistency that we want our network to learn and hence would have to be explicitly detected and removed; using single-take shots eliminates this problem.

For the driving scenarios, we use KITTI-RAW [22] and Malaga [3], the first of which contains several sequences and the second of which contains one long shot of camera footage recorded from a driving car. For each sequence, we use around

1 % of frames as validation, sampled from the beginning, center, and end of the sequence and sampled such that they do not overlap the actual training data. The only difference is KITTI-RAW, which is already broken up into sequences. Here, we sample full sequences to go either to the training or validation set, and use 10 % for the validation set. This ensures that the validation set contains approximately the same amount of frames from driving and movie-like scenarios. Table 2 summarizes the datasets used and the amount of frames from each.

In total, we thus have approximately 464K training frames. However, in movie sequences, the camera and object motions are often small. Therefore, during both training and validation, we predict each frame twice, once from the adjacent frames as shown in Fig. 3, and once with doubled spacing between the frames. Therefore, a target frame at time t provides two training samples, one where it is predicted from the frames at $t-3, t-1, t+1$, and $t+3$, and one where it is predicted from $t-6, t-2, t+2$, and $t+6$. This ensures that we have a sufficient amount of large motions in our frame interpolation training set. In total, our training and validation sets contain 928,410 and 16,966 samples, respectively.

Training details. As shown in Fig. 3, each training sample of our network consists of the four input frames and the center frame which the network should predict. During training, each quadruple of frames is separately normalized by subtracting the mean of the four input frames and dividing by their standard deviation. We found this to work better than normalization across the full dataset. We believe the reason for this is that in correspondence estimation, it is more important to consider the structure *within* a sample than the structure across samples, the later of which is important for classification tasks. To put it differently, whether a light bar or a dark bar moves to the right does not matter for optical flow and should produce the same output.

As data augmentation, we randomly crop the input images to rectangles with the aspect ratio 2:1, and resize the cropped images to a resolution of 384×192 pixel, resulting in randomization of both scale and translation. For all input frames belonging to a training sample, we use the same crop. Furthermore, we randomly flip the images in horizontal and vertical direction, and randomly switch between forward and backward temporal ordering of the input images. We use a batch size of 8, train our network using ADAM and use a loss consisting of a structural similarity loss (SSIM) [33] and an L_1 loss, weighted equally. The initial learning rate is set to $1e-4$, and halved after 3, 6, 8, and 10 epochs. We train our network for 12 epochs; after this point, we did not notice any further decrease in our loss on the validation set.

4 From interpolation to Optical Flow

Given a frame interpolation network, it has been shown before [18] that motion is learned by the network and can be extracted. However, this only works for regions with sufficient texture. In unstructured areas of the image, the photometric error that is used to train the frame interpolation is not informative, and even a wrong motion estimation can result in virtually perfect frame reconstruction.

Table 2: Training data for interpolation.

Source	Type	Training frames	Validation frames
Birdman	Full movie	155,403	1,543
Victoria	Full movie	188,700	1,878
KITTI-RAW	Driving	39,032	3,960
Malaga	Driving	51,285	460
Atonement	Movie clip	7,062	44
Children of Men	Movie clip	9,165	65
Baby Driver	Movie clip	3,888	14
True Detective	Movie clip	8,388	57
Total		464,205	8,483

What is missing for good optical flow estimation, then, is the capability to group the scene and to fill in the motion in unstructured regions, *ie.* to address the aperture problem. Furthermore, the mechanism used to extract the motion in [18] is slow, since it requires a complete backpropagation pass for each correspondence. To effectively use frame interpolation for optical flow computation, two steps are thus missing: (a) to add knowledge about grouping and region fill-in to a network that can compute correspondences, and (b) to modify the network to directly yield an optical flow field, making expensive backpropagation steps unnecessary at test time. Luckily, both objectives can be achieved by fine-tuning the network to directly estimate optical flow, using only a very limited amount of annotated ground truth data.

For this, we replace the last layer of the `Dec1` block with a vanilla 3×3 convolutional layer with two output channels instead of one, and train this network using available ground truth training data, consisting of the training sets of KITTI-2012 [8], KITTI-2015 [22] and the clean and final passes of MPI-Sintel [4], for a total of about 2500 frames. We use 10 % of the data as validation.

We again use ADAM with an initial learning rate of 10^{-4}, halve the learning rate if the error on the validation set has not decreased for 20 epochs, and train for a total of 200 epochs using the endpoint error as the loss. Except for the temporal reversal, we use the same augmentations as described above.

As we will see in the next section, this simple fine-tuning procedure results in a network that computes good optical flow, and even outperforms networks with comparable architecture that were trained using large amounts of synthetically generated optical flow.

5 Experiments

In this section, we demonstrate the effectiveness of our method for interpolation and optical flow estimation, and provide further experiments showing the importance of pre-training and the effects of reduced ground truth data for fine-tuning.

Fig. 4: Visual interpolation results (unseen data). From top to bottom: linear blending; interpolation using [23], $\mathcal{L}_F$ variant; interpolation using our method; ground truth. While the results from [23] are sharper, they produce significant artifacts, for example the tree in the right example. Our method tends to be blurrier, but captures the gist of the scene better; this is reflected in the superior quantitative results.

5.1 Temporal interpolation

To evaluate the interpolation performance of our network, we compare with [23], a state-of-the-art method for frame interpolation. Unlike ours, [23] is specifically designed for this task; in contrast, we use a standard hourglass network. We compute temporal interpolations for 2800 frames from natural movies, a synthetic movie (Sintel), and driving scenarios. All frames were not previously seen in training. To compute interpolated color frames, we simply run our network once for each input color channel. Table 3 shows results on both PSNR and SSIM. For [23], we report both the $\mathcal{L}_1$ and $\mathcal{L}_F$ results; according to [23], the former is better suited for numerical evaluation, while the later produces better visual results. We outperform both variants in both metrics.

Table 3: Interpolation performance in PSNR (SSIM).

	Real movie	Synthetic movie	Driving	All
[23], $\mathcal{L}_1$	33.15 (0.915)	25.73 (0.841)	18.26 (0.664)	28.80 (0.854)
[23], $\mathcal{L}_F$	32.98 (0.911)	25.44 (0.825)	18.04 (0.631)	28.59 (0.843)
Ours	**34.68** (**0.928**)	**26.46** (**0.859**)	**19.76** (**0.710**)	**30.13** (**0.8741**)

Figure 4 shows quantitative results of the interpolation for all three scenarios. Visually, the interpolation results are good. In particular, our method can handle

large displacements, as can be seen in the helmet strap in Fig 4, first column, and the tree in Fig. 4, third column. The results of [23] tend to be sharper; however, this comes with strong artifacts visible in the second row of Fig. 4. Both the helmet strap and the tree are not reconstructed significantly better than when using simple linear blending (first row); our method, while slightly blurry, localizes the objects much better.

5.2 Optical Flow

Results on benchmarks. To demonstrate the effectiveness of our method, dubbed *IPFlow* for Interpolation Pretrained Flow, we test the optical flow performance on the two main benchmarks, KITTI [8, 22] and MPI-Sintel [4]. Since our method uses four input frames, we double the first and last frames to compute the first and last flow field within a sequence, respectively, thereby obtaining flow corresponding to all input frames. Furthermore, like FlowNet [6], we perform a variational post-processing step to remove noise from our flow field. Computing the flow on a NVIDIA M6000 GPU takes 60 ms for the CNN; the variational refinement takes 1.2 seconds.

Table 4: Quantitative evaluation of our method.

	Sintel		Kitti-2012	Kitti-2015
	Clean	Final		
Supervised methods				
FlowNet2-ft [12]	**4.16**	**5.74**	**1.8**	**11.48 %**
FlowNetS+ft+v [6]	6.16	7.22	9.1	
SpyNet+ft [24]	6.64	8.36	4.1	35.07 %
Un- and semisupervised methods				
DSTFlow [25]	10.41	11.28	12.4	39 %
USCNN [1]		8.88		
Semi-GAN [15]	6.27	7.31	6.8	31.01 %
UnFlow-CSS [21]	9.38	10.22	**1.7**	**11.11 %**
IPFlow (ours)	**5.95**	**6.59**	3.5	29.54 %
IPFlow-Scratch	8.35	8.87		

Table 4 shows the errors on the unseen test sets (average endpoint error for Sintel and KITTI-2012, `Fl-All` for KITTI-2015); Figure 5 shows qualitative results. While we do not quite reach the same performance as more complicated architectures such as FlowNet2 [12] or UnFlow-CSS [21][3], on all datasets we outperform methods which are based on simple architectures comparable to ours, including those that were trained with large amounts of annotated ground

[3] For UnFlow, test set results are only available for the -CSS variant, which is based on a FlowNet2 architecture. The simpler UnFlow-C is not evaluated on the test sets.

Fig. 5: Visual results. From top to bottom: Input image; Ground truth flow; Result of IPFlow; Training from scratch. The flow computed using pure training from scratch is reasonable, but using pre-training yields significantly better optical flow maps.

truth data (FlowNetS+ft+v [6] and SpyNet [24]). For these simple architectures, pre-training using a slow-motion task is hence superior to pre-training using synthetic, annotated optical flow data. UnFlow-CSS is the only method outperforming ours on KITTI that does not require large amounts of annotated frames; yet, they use a considerably more complicated architecture and only achieve state-of-the-art results in driving scenarios and not on Sintel.

Performance without pretraining. To evaluate whether the Sintel training data might be enough to learn optical flow by itself, we also tried training our network from scratch. We test two training schedules, first using the same learning parameters as for the fine-tuning, and second the well-established *s_short* schedule from [12]. As shown in Fig. 6, the network is able to learn to compute optical flow even without pre-training, and benefits from using the *s_short* schedule[4]. However, at convergence the error of the network without pre-training on unseen validation data is around 50 % higher. This is also visible in Table 4, where *IPFlow-Scratch* denotes the training from scratch using *s_short*; again, the errors are considerably higher. Thus, important properties of motion must have been learned during the unsupervised pretraining phase.

Using a low number of fine-tuning frames. As we just showed, using only the training set and no fine-tuning results in significantly worse performance; pre-training from interpolation is clearly beneficial. However, this now points to another, related question: Once we have a pre-trained network, how much annotated training data is actually required to achieve good performance? In other words, does pre-training free us from having to annotate or generate thousands of ground truth optical flow frames, and if so, how large is this effect?

We tested this question by repeating the finetuning procedure using only a small amount (25-200) of randomly chosen frames from the respective training sets. Since the scenario for using very few annotated frames points to application-

[4] For fine-tuning after pretraining, *s_short* gives higher errors than our schedule.

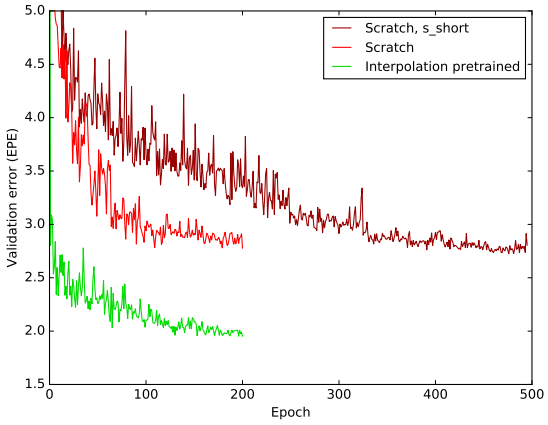

Fig. 6: Using interpolation as pre-training, the network learns to adapt to optical flow. Flow can also be learned without pre-training, but in this case the error is 50 % higher.

specific optical flow (for example, flow specifically for driving), we perform this experiment separately for different datasets, KITTI (containing both KITTI-2015 [22] and KITTI-2012 [8]), Sintel (clean) and Sintel (final). All trained networks are tested on the same validation set for the respective dataset, and we repeated the experiment three times and averaged the results.

Figure 7 shows the results. While using only 25 frames is generally not enough to estimate good optical flow, the performance quickly improves with the number of available training frames. After seeing only 100 training frames, for all datasets the performance is within 0.5 px EPE of the optimal performance achievable when using the full training sets for the respective dataset. This shows that a interpolation-pretrained network such as the one presented here can be quickly and easily tuned for computing flow for a specific application, and does not require a large amount of annotated ground truth flow fields.

6 Conclusion

In this work, we have demonstrated that a network trained for the task of interpolation does learn about motion in the world. However, this is only true for image regions containing sufficient texture for the photometric error to be meaningful. We have shown that, using a simple fine-tuning procedure, the network can be taught to (a) fill in untextured regions and (b) to output optical flow directly, making it more efficient than comparable, previous work [18]. In particular, we have shown that only a small number of annotated ground truth optical flow frames is sufficient to reach comparable performance to large datasets; this provides the user of our algorithm with the choice of either increasing the accuracy of the optical flow estimation, or to require only a low number of annotated ground truth frames. Demonstrating the importance of pre-training, we have

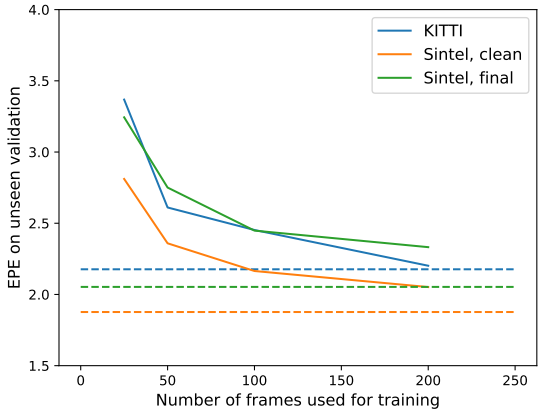

Fig. 7: Fine-tuning with a small amount of frames. With only 100 frames, the performance on all validation sets gets within $0.5px$ EPE of the optimal performance. The dashed lines show the performance when using the full training set for each dataset.

shown that the same network without the interpolation pre-training performs significantly worse; our network also outperforms all other methods with comparable architectures, regardless whether they were trained using full supervision or not. As a side effect, we have demonstrated that, given enough and sufficiently varied training data, even a simple generic network architecture outperforms a specialized architecture for frame interpolation.

Our work suggests several directions for future work. First, it shows the usefulness of this simple proxy task for correspondence estimation. In the analysis of still images, however, we often use a proxy task that requires some *semantic* understanding of the scene, in the hope of arriving at internal representations of the image that mirror the semantic content. As video analysis moves away from the pixels and more towards higher-level understanding, finding such proxy tasks for video remains an open problem. Second, even when staying with the problem of optical flow estimation, we saw that optimized pipelines together with synthetic training data still outperform our method. We believe, however, that even those algorithms could benefit from using a pre-training such as the one described here; utilizing it to achieve true state-of-the-art performance on optical flow remains for future work.

Lastly, the promise of unsupervised methods is that they scale with the amount of data, and that showing more video to a method like ours would lead to better results. Testing how good an interpolation (and the subsequent flow estimation) method can get by simply watching more and more video remains to be seen.

Acknowledgements & Disclosure. JW was supported by the Max Planck ETH Center for Learning Systems. MJB has received research funding from Intel, Nvidia, Adobe, Facebook, and Amazon. While MJB is a part-time employee of Amazon, this research was performed solely at, and funded solely by, MPI.

References

1. Ahmadi, A., Patras, I.: Unsupervised convolutional neural networks for motion estimation. In: 2016 IEEE International Conference on Image Processing (ICIP). pp. 1629–1633 (Sept 2016). https://doi.org/10.1109/ICIP.2016.7532634
2. Alletto, S., Abati, D., Calderara, S., Cucchiara, R., Rigazio, L.: Transflow: Unsupervised motion flow by joint geometric and pixel-level estimation. Tech. rep., arXiv preprint arXiv:1706.00322 (2017)
3. Blanco, J.L., Moreno, F.A., Gonzalez-Jimenez, J.: The malaga urban dataset: High-rate stereo and lidars in a realistic urban scenario. International Journal of Robotics Research **33**(2), 207–214 (2014). https://doi.org/10.1177/0278364913507326
4. Butler, D., Wulff, J., Stanley, G., Black, M.: A naturalistic open source movie for optical flow evaluation. In: ECCV (2012)
5. Doersch, C., Gupta, A., Efros, A.A.: Unsupervised visual representation learning by context prediction. In: ICCV (2015)
6. Dosovitskiy, A., Fischer, P., Ilg, E., Hausser, P., Hazirbas, C., Golkov, V., van der Smagt, P., Cremers, D., Brox, T.: Flownet: Learning optical flow with convolutional networks. In: International Conference on Computer Vision (ICCV) (2015)
7. Fragkiadaki, A., Seybold, B., Sukthankar, R., Vijayanarasimhan, S., Ricco, S.: Self-supervised learning of structure and motion from video. In: arxiv (2017) (2017)
8. Geiger, A., Lenz, P., Stiller, C., Urtasun, R.: Vision meets robotics: The KITTI dataset. International Journal of Robotics Research **32**(11), 1231–1237 (Sep 2013). https://doi.org/10.1177/0278364913491297
9. Godard, C., Mac Aodha, O., Brostow, G.J.: Unsupervised monocular depth estimation with left-right consistency. In: CVPR (July 2017)
10. Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., Bengio, Y.: Generative adversarial nets. In: NIPS (2014)
11. Güney, F., Geiger, A.: Deep discrete flow. In: ACCV (2016)
12. Ilg, E., Mayer, N., Saikia, T., Keuper, M., Dosovitskiy, A., Brox, T.: Flownet 2.0: Evolution of optical flow estimation with deep networks. In: CVPR (2017)
13. Jaderberg, M., Simonyan, K., Zisserman, A., kavukcuoglu, k.: Spatial transformer networks. In: NIPS (2015)
14. Janai, J., Güney, F., Wulff, J., Black, M., Geiger, A.: Slow flow: Exploiting high-speed cameras for accurate and diverse optical flow reference data. In: CVPR (2017)
15. Lai, W.S., Huang, J.B., Yang, M.H.: Semi-supervised learning for optical flow with generative adversarial networks. In: NIPS (2017)
16. Larsson, G., Maire, M., Shakhnarovich, G.: Colorization as a proxy task for visual understanding. In: CVPR (2017)
17. Liu, C., Freeman, W.T., Adelson, E.H., Weiss, Y.: Human-assisted motion annotation. In: CVPR (2008)
18. Long, G., Kneip, L., Alvarez, J.M., Li, H., Zhang, X., Yu, Q.: Learning Image Matching by Simply Watching Video (2016)
19. Mayer, N., Ilg, E., Hausser, P., Fischer, P., Cremers, D., Dosovitskiy, A., Brox, T.: A large dataset to train convolutional networks for disparity, optical flow, and scene flow estimation. In: CVPR (2016)
20. Meinhardt, T., Moller, M., Hazirbas, C., Cremers, D.: Learning proximal operators: Using denoising networks for regularizing inverse imaging problems. In: ICCV (2017)

21. Meister, S., Hur, J., Roth, S.: Unflow: Unsupervised learning of optical flow with a bidirectional census loss. arXiv preprint arXiv:1711.07837 (2017)
22. Menze, M., Heipke, C., Geiger, A.: Discrete optimization for optical flow. In: German Conference on Pattern Recognition (GCPR). vol. 9358, pp. 16–28. Springer International Publishing (2015)
23. Niklaus, S., Mai, L., Liu, F.: Video frame interpolation via adaptive separable convolution. In: ICCV (2017)
24. Ranjan, A., Black, M.J.: Optical flow estimation using a spatial pyramid network. Tech. rep., arXiv (2016)
25. Ren, Z., Yan, J., Ni, B., Liu, B., Yang, X., Zha, H.: Unsupervised deep learning for optical flow estimation. In: AAAI Conference on Artificial Intelligence (2017)
26. Richter, S.R., Hayder, Z., Koltun, V.: Playing for benchmarks. In: ICCV (2017)
27. Sedaghat, N., Zolfaghari, M., Brox, T.: Hybrid learning of optical flow and next frame prediction to boost optical flow in the wild. Tech. rep., arXiv:1612.03777 (2017), https://arxiv.org/abs/1612.03777
28. Sun, C., Shrivastava, A., Singh, S., Gupta, A.: Revisiting unreasonable effectiveness of data in deep learning era. In: ICCV (2017)
29. Sun, D., Roth, S., Black, M.: A quantitative analysis of current practices in optical flow estimation and the principles behind them. International Journal of Computer Vision **106**(2), 115–137 (2014). https://doi.org/10.1007/s11263-013-0644-x
30. Sun, D., Sudderth, E., Black, M.J.: Layered segmentation and optical flow estimation over time. In: CVPR (2012)
31. Sun, D., Yang, X., Liu, M.Y., Kautz, J.: Pwc-net: Cnns for optical flow using pyramid, warping, and cost volume. arXiv preprint arXiv:1709.02371 (2017)
32. Vondrick, C., Pirsiavash, H., Torralba, A.: Anticipating visual representations from unlabeled video. In: CVPR (2016)
33. Wang, Z., Bovik, A.C., Sheikh, H.R., Simoncelli, E.P.: Image quality assessment: from error visibility to structural similarity. IEEE Transactions on Image Processing **13**(4), 600–612 (April 2004). https://doi.org/10.1109/TIP.2003.819861
34. Xu, J., Ranftl, R., Koltun, V.: Accurate optical flow via direct cost volume processing. In: CVPR (2017)
35. Yu, J.J., Harley, A.W., Derpanis, K.G.: Back to Basics: Unsupervised Learning of Optical Flow via Brightness Constancy and Motion Smoothness (2016)
36. Zhou, T., Brown, M., Snavely, N., Lowe, D.G.: Unsupervised learning of depth and ego-motion from video. In: CVPR (2017)