FlowNet: Learning Optical Flow with Convolutional Networks

Hauptseminar Recent Advances in Computer Vision

Sharon Anna Thomas

Abstract—Convolutional neural networks (CNNs) are widely used in computer vision applications due to their successes in the field. Recent works, including this paper, have applied CNNs to predict optical flow. This supervised learning task is carried out by using two architectures: a generic one where the network learns to process the data by itself and one that includes a correlation layer to map corresponding feature vectors. They are trained using a generated synthetic Flying Chairs dataset since available ground-truth datasets are not large enough. These architectures show a competitive accuracy of 5 to 10 fps when compared with other well-performing methods and also sufficiently generalize to existing datasets such as Sintel and KITTI.

1 Introduction

Convolutional neural networks have become a preferred method to solve many computer vision tasks such as classification, per-pixel predictions like semantic segmentation, or depth estimation. In this paper[6], the authors have proposed training CNNs to estimate optical flow from a pair of images. This task requires learning feature representation and finding correspondences between the two input images to obtain a fairly accurate estimation of the optical flow. Although an exclusive, separate correlation layer was developed for feature matching in the initial stages, it was later found that it wasn't necessary, as the raw network itself performed relatively well in the task. When training a CNN end-to-end, a large dataset with appropriate ground truth flows was needed. Since the existing datasets were too small despite data augmentation, a synthetic dataset called Flying Chairs dataset was generated which involved overlaying segmented images of chairs from[1] on random background images from Flickr.

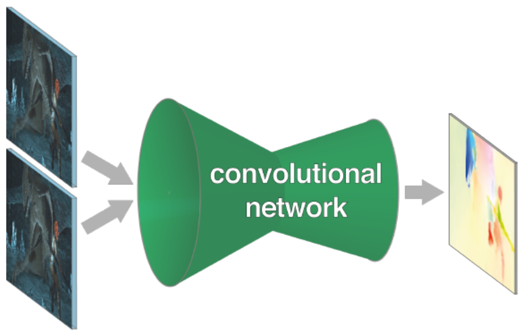


Fig. 1. FlowNets estimate optical flow. Information is initially compressed in the contractive part and then refined to higher resolution in the expanding part of the network [6].

2 Background

2.1 Convolutional Neural Networks (CNN)

Convolutional neural networks are a class of deep neural networks that are commonly used to analyse visual imagery[17]. They consist of four main operations:

1. Convolution
2. Pooling
3. ReLU
4. Fully Connected Layers

Institut für Visualisierung und Interaktive Systeme, Universität Stuttgart. Sommersemester 2020.

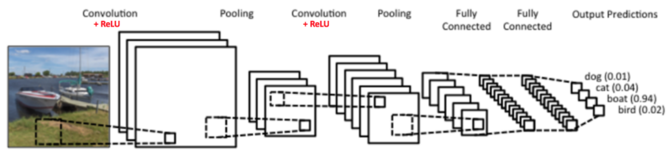


Fig. 2. Example of a convolutional neural network used for classification [17].

An image is a matrix of pixel values. Color images have three channels (three matrices) - red,blue and green while grey-scale images are a 2D matrix. Each matrix element, in both cases, have a range of pixel values from 0 to 255 indicating the intensity of the corresponding pixel. For example: 0 indicates black and 255 indicates white in grey-scale images. The above mentioned four operations are used to extract relevant information from the given input and find meaningful output from it.

2.1.1 Convolution

This operation extracts features from the input image. Consider a 6x6 matrix and a 3x3 matrix as shown in Figure. The 3x3 matrix is called a kernel. The element-wise multiplication is computed between the 2 matrices and the outputs are added to get the final convolved feature. The kernel matrix is slid over the image by a value stride. The kernel or the filter acts as the feature detector for the image. Different filters provide different feature maps for the same input image. The filters can detect edges, curves, etc depending on the values provided in the filter.

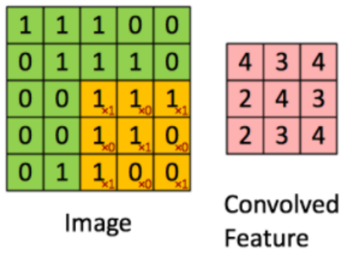
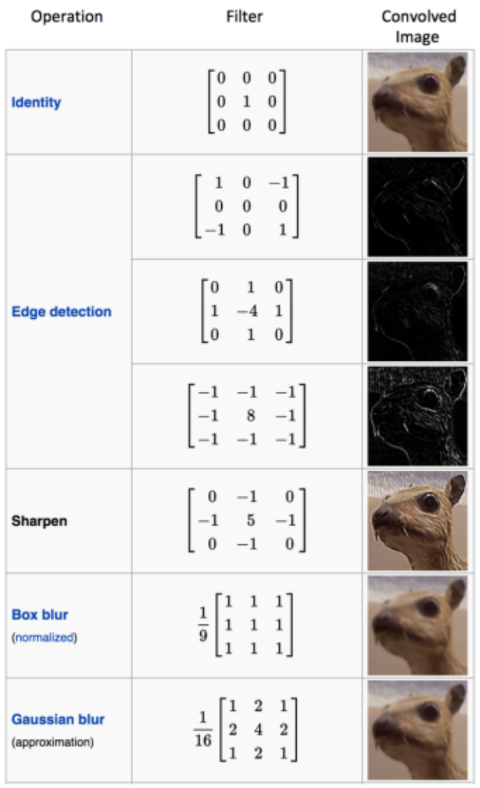


Fig. 3. Convolution operation. The orange matrix (filter) is slid across the green matrix(original image) to obtain the pink matrix (convolved feature) [17].

2.1.2 ReLU

ReLU stands for rectified linear unit and introduces non-linearity into the convolutional network. It is an element-wise operation and sets all



Operation	Filter	Convolved Image
Identity	$\begin{bmatrix}0&0&0\\0&1&0\\0&0&0\end{bmatrix}$	
Edge detection	$\begin{bmatrix}1&0&-1\\0&0&0\\-1&0&1\end{bmatrix}$	
	$\begin{bmatrix}0&1&0\\1&-4&1\\0&1&0\end{bmatrix}$	
	$\begin{bmatrix}-1&-1&-1\\-1&8&-1\\-1&-1&-1\end{bmatrix}$	
Sharpen	$\begin{bmatrix}0&-1&0\\-1&5&-1\\0&-1&0\end{bmatrix}$	
Box blur (normalized)	$\frac{1}{9}\begin{bmatrix}1&1&1\\1&1&1\\1&1&1\end{bmatrix}$	
Gaussian blur (approximation)	$\frac{1}{16}\begin{bmatrix}1&2&1\\2&4&2\\1&2&1\end{bmatrix}$	

Fig. 4. Examples of convolved features using various filters on a given input image [17].

negative pixel values in the feature map by zero. Example can be seen in figure 5. Some other non-linear functions such as tanh or sigmoid can be used as well depending on the application.

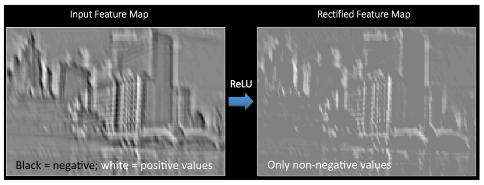


Fig. 5. The resultant image after ReLU operation [17].

2.1.3 Pooling

Spatial pooling helps in reducing the size of the feature map by downsampling but still retains the most important information. Pooling can be of different types such as Max, Average, Sum, etc. For example: Max pooling involves taking the largest element from the rectified feature map within a window of specific size, as shown in the figure.

2.1.4 Fully Connected Layers

When the layers are fully connected, each neuron in the previous layer will be connected to a neuron in the next layer. This provides a nonlinear combination of the features which are usually much better at approximating the output.

2.2 Optical Flow

Optical flow is the perceived motion of objects across consecutive frames, caused by the relative movement of the object and camera [8]. For example: 5 frame clip of ball (Figure 7) moving from bottom left to top right. The problem of optical flow can be expressed as:

$$I(x,y,t) = I(x+\delta x, y+\delta y, t+\delta t)$$

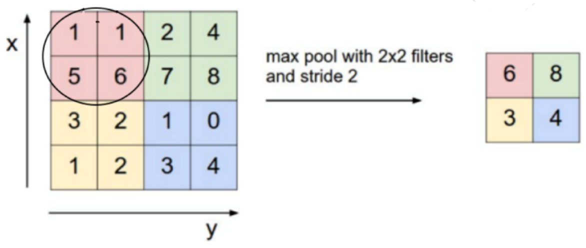


Fig. 6. Max pooling operation [17].

where $I(x,y,t)$ is image intensity expressed as a function of space (x,y) and time (t). If the pixels of the first image $I(x,y,t)$ is moved by $(x,\delta y)$ over δt time, then a new image $I(x+\delta x, y+\delta y, t+\delta t)$ is obtained. Optical flow has also played a prominent role in the entertainment industry where it is used to slow down parts of video footage, especially in cases of low frame rates.

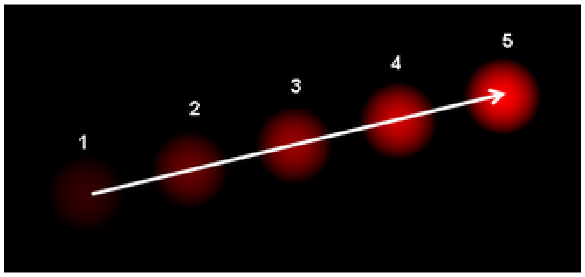


Fig. 7. Optical flow of a ball across multiple frames. The optical flow vector can also be extracted from the sequence of images [8].

3 Related Work

Optical Flow. Optical flow estimation has mostly been based on the variational approach introduced by Horn and Schunck[10]. DeepFlow[18] and DeepMatching are similar to FlowNets in the sense that these methods also use information aggregation using convolution and pooling. Some authors have applied machine learning to optical flow, such as Sun *et al.* [16] has detailed the statistics of optical flow and others have used classifiers to choose from different inertial estimates or to gather probabilities of occlusion occurring [6]. Research on unsupervised learning include using autoencoders[11] called 'synchrony autoencoder'. However, these methods have been proved inefficient when compared to classical methods on realistic videos.

Convolutional Networks. Convolutional neural networks are classically applied on classification tasks. There has been research on per-pixel predictions such as semantic segmentation, depth prediction and edge detection. Building on this, a simple approach to estimating optical flow could have involved applying a CNN in 'sliding window' fashion and calculating a prediction for each patch in the input, albeit leading to high computational costs. The approach used in this paper integrates the work of two papers - Long *et al.* [13] and Dosovitskiy *et al.*[7] where the coarse feature maps are refined using 'upconvolutional' layers. Additionally this paper [6] 'upconvolves' the entire feature maps and concatenates it with the 'contractive' part of the network, thus allowing for the transfer of high-level information as well.

4 Network Architectures

When provided with an ample labelled input data, convolutional neural networks are able to learn the input-output relations with relative accuracy. The authors of the paper have chosen to employ this end-to-end learning approach; where the network is given input data, which consists of image pairs and ground truth flows, the

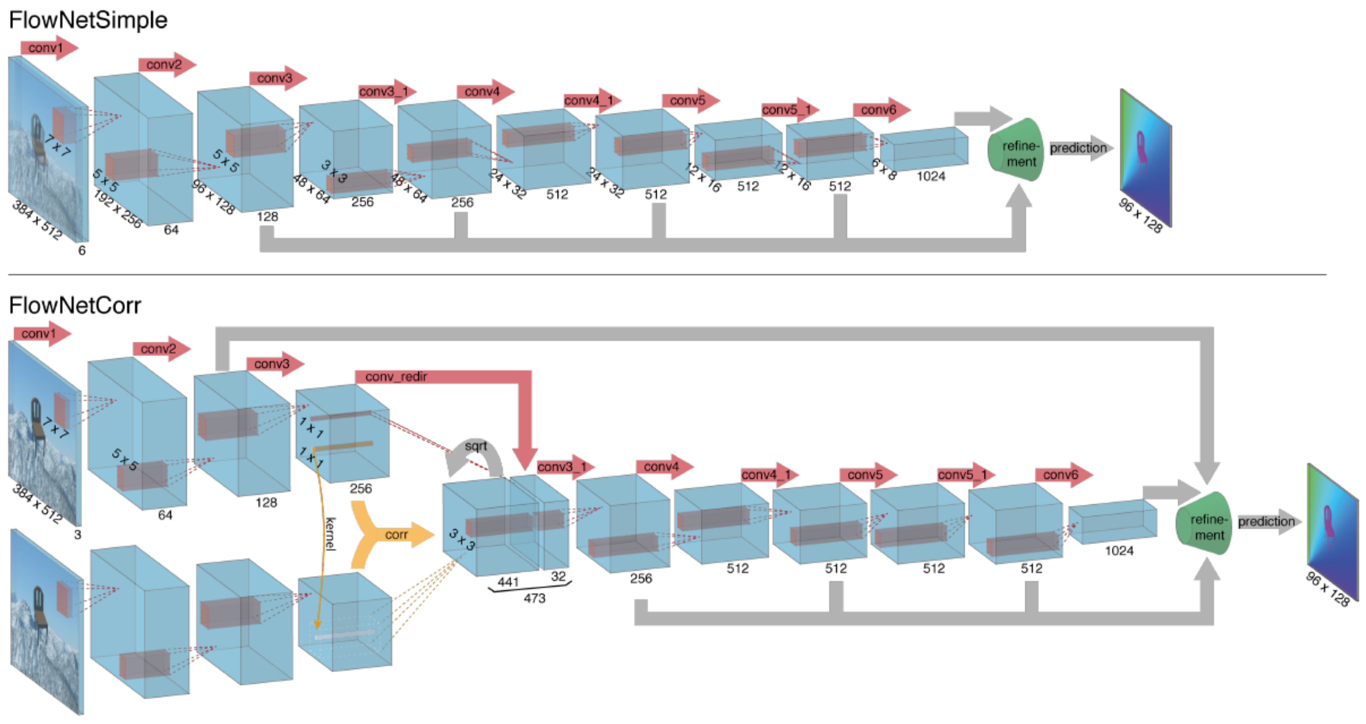


Fig. 8. FlowNetSimple (top) and FlowNetCorr (bottom). The green funnel represents the flow refinement operation [6].

x-y optical flows are directly predicted. Pooling in CNNs allows for computationally feasible network training and also aggregates information resulting in a lower resolution image (contracting part). However, for a dense-per-pixel representation, this needs to be refined through the addition of an expanding part. Both the contracting and expanding parts are trained using propagation. The architectures are shown in figure 8.

Contracting part. The simplest approach is to stack both images together and feed it into the network as input. This network is rather generic and consists almost entirely of only convolutional layers. The network itself decides how to process and extract the required optical flow estimation and is called FlowNetSimple.

A secondary approach is to create two separate and identical streams for the two images and combine them later as show in Figure 8 (bottom). This requires the network to produce features from both input images and 'match' between the feature vectors. A 'correlation layer' is added to enable multiplicative patch comparisons between the two feature maps. Given two multi-channel feature maps f_1, f_2 with w, h and c being the width, height and number of channels, the correlation layer compares each patch from f_1, with each patch from f_2. [6]

Considering a single comparison, correlation of two patches centred at x_1 in the first map and at x_2 in the second map is defined as:

$$c(x_1,x_2) = \sum \langle f_1(x_1+o), f_2(x_2+o)\rangle \tag{1}$$

where $o\varepsilon[-k,k]x[-k,k]$ for a square patch of size $K := 2k+1$. Computing $c(x_1,x_2$ requires $c.K^2$ multiplications giving a large result and leading to inefficient forward and backward passes. To counter this, the authors limit the maximum displacement for comparisons and implement striding in both feature maps.

For maximum displacement d and each location $x1$, $c(x_1,x_2)$ is computed only in a neighbourhood of size $D := 2d+1$, thus limiting the range of x_2. Strides s_1 and s_2 are used to restrict x_2 within the neighbourhood centred around x_1. Although in theory. the obtained result is four-dimensional: for every combination of two 2D positions, the scalar product of the two vectors is calculated, the practical approach was to organize the relative displacements in channels and obtain an output of size $(w.h.D^2)$ [6].

Expanding part To refine the flow to higher resolution, 'Upconvolution' is performed which involves 'unpooling' and a convolution. The 'upconvolution' is applied to the feature map and it is then concatenated with feature maps from the contractive part of the network. Thus both high-level information from coarse feature maps and fine local information from lower layer feature maps are preserved. Each repetition doubles the resolution. This layer is shown in Figure 9. An available alternative to this method is bilinear upsampling to full image resolution.

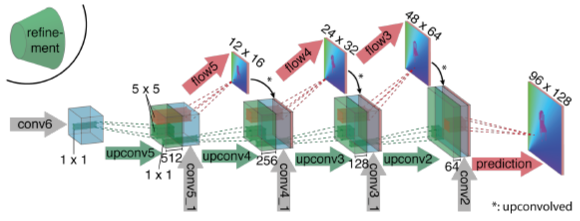


Fig. 9. Low resolution flow is refined to a higer resolution using upconvolution [6].

Variational Refinement A variational approach for refinement could involve using the method mentioned in [4]. Starting at 4 times downsampled resolution and then using coarse to fine scheme with 20 iterations, would bring the flow field to full image resolution. Image boundaries are computed using the approach from [12] and replace the smoothness coefficient by $\alpha = exp(-\lambda b(x,y)^\kappa)$ where $b(x,y)$ is the thin boundary strength. Though this method is computationally more expensive than bilinear upsampling, a smooth and sub-pixel accurate flow field can be obtained. The results obtained in this way are denoted by a '+v' suffix. An example of variational refinement is shown in Figure 10.

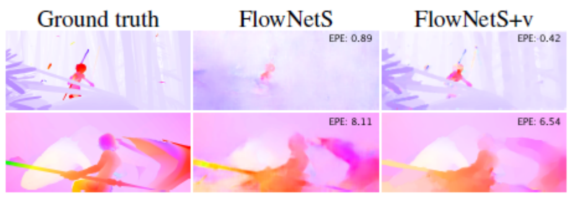


Fig. 10. Variational refinement. Small motions result in much better predictions (first row). Larger motions has errors but the flow field is smoothed, hence the lower EPE [6].

5 Training Data

The input data must contain ground truth flows to learn to perform the task. However, obtaining the ground truth flow is relatively hard since pixel correspondences cannot be easily determined. Available datasets are summarized in the Table 1 :

Tabelle 1. Summary of available datasets along with the synthetic Flying Chairs [6]:

	Frame Pairs	Frames with ground truth	Ground truth density per frame
Middlebury	72	8	100%
KITTI	194	194	50%
Sintel	1041	1041	100%
Flying Chairs	22872	22872	100%

5.1 Existing Datasets

The Middlebury dataset [2] has 8 image pairs with ground truth flows generated using four different techniques. The displacements are small and mostly below 10 pixels.

The KITTI dataset [9] is larger containing 194 image pairs with large displacements. The ground truth flow is obtained by recording a real world scene simultaneously with a camera and a 3D laser scanner. This leads to the assumption that the motion is caused by the moving observer and the scene itself is rigid. However, the dataset contains sparse optical flow ground truth.

The MPI Sintel [5] dataset has rendered artificial scenes to obtain the ground truth flow and is the largest available. There are two versions containing 1041 training images each: Final version which contains motion blur and atmospheric effects such as fog and the Clean version which does not contain these effects. The dataset has dense ground truth for small and large displacement magnitudes.

5.2 Flying Chairs

The Sintel dataset does not contain enough data to train the large networks. Instead, Flying Chairs dataset was created by applying affine transformations to 3D chair models obtained from Flickr and other publicly available set of renderings (Figure 11).

2D affine transformation parameters are randomly sampled to generate motion for the background and the chairs. Since the motion of the chairs are relative to the background, the flow can be interpreted as both the camera and the objects moving. The transformation parameters are used to generate the second image, the ground truth optical flow and occlusion regions. This generated dataset has 22,872 image pairs and flow fields.

5.3 Data Augmentation

Data augmentation is crucial to improve the generalization of the neural networks and to avoid overfitting, despite the fairly large Flying Chairs dataset. This is done online during training and on a GPU to be reasonably quick. The augmentations used include geometric transformations such as translation, rotation and scaling. Additive Gaussian noise and other changes in brightness, contrast, gamma and color are also applied. In order to increase the variety of flow fields as well as variety of images, the transformation is applied to both pairs, in addition to a smaller relative transformation between the two images. For example: translation can be sampled from the range [-20%,20%], rotation from [-17%,17%] etc. Additive brightness changes can be done using a Gaussian with sigma 0.2.

6 Experiments

The results of the networks on Sintel, KITTI, Middlebury and Flying Chairs dataset are detailed below. Fine-tuning on Sintel data and variational refinement of the predicted flow fields were also carried out.

6.1 Network and Training Details

The architecture of the different networks is kept consistent: it has nine convolutional layers with a stride of 2 in six of the layers. ReLU non-linearity is added after each layer and the layers are not fully connected allowing for input images of arbitrary size. The convolutional filter sizes decrease towards the deeper layers: 7x7 for the first, 5x5 for the next two and 3x3 for the rest. The number of feature maps roughly doubles down the network with a stride of 2. The parameters chosen for the correlation layer in FlowNetC are as follow: $k = 0, d = 20, s_1 = 1, s_2 = 2$. The standard optical flow error measure ,endpoint error (EPE) is used as the loss function. It is the Euclidean distance between the predicted flow vector and the ground truth, averaged over all pixels.

A modified caffe framework is used to train the CNNs and Adam[14] is used as the optimization method. Adam parameters are fixed at $\beta_1 = 0.9$ and $\beta_2 = 0.999$. Mini batch training of 8 image pairs is used since each pixel is considered a training sample. Initial learning rate is $\lambda = 1e-4$. After the 300k iterations, the rate is divided by 2 every 100k iterations. To counter the exploding gradient issue in FlowNetCorr with the initial learning rate, λ is set to $1e-6$ and then gradually increased to $1e-4$ after 10k iterations.

The Flying Chairs dataset is split into 22,232 training and 640 test samples and the Sintel into 908 training and 133 validation pairs to monitor overfitting during training and fine-tuning.

Fine-tuning Since the datasets are varied in object types and motions, the networks are fine-tuned on the target datasets. Images from both Final and Clean versions of the Sintel training set is used with a low-learning rate of $\lambda = 1e-6$. The fine-tuned networks are denoted by a '+ft' suffix.

6.2 Results

Table 2 shows the endpoint error (EPE) of the FlowNet networks and other well-performing networks on Sintel, KITTI, Middlebury and Flying Chairs. The runtimes of the different methods on Sintel are also shown.

The networks trained on the synthetic Flying Chairs dataset perform much better when compared to LDOF[4] method. Additional fine-tuning on Sintel, improved their performances against EPPM[3] on Sintel Final and KITTI.

Sintel. On Sintel Clean, FlowNetC performs better than FlowNetS, but not in the case of Sintel Final. Figure 12 shows a few examples of the FlowNets compared to ground truth and EpicFlow[15]. Though the networks produce appealing results in terms of visuals, the EPE is still larger since EPE prefers over-smoothed solutions. This is more obvious in images with largely smooth background regions - the output however is noisy and non-smooth. Variational refinement can

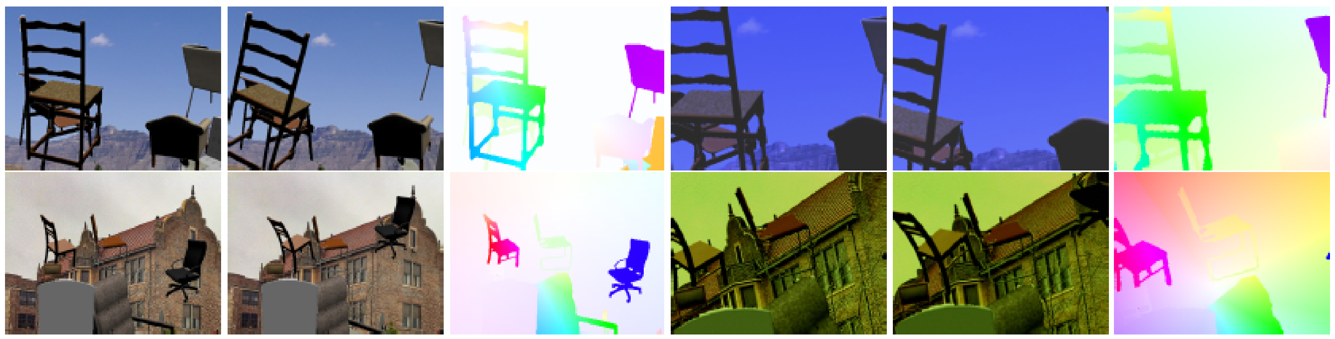


Fig. 11. Samples from the Flying Chairs dataset. Image pair and color coded flow field are shown in the first three columns. Augmented image pairs in the last three columns [6].

Method	Sintel Clean		Sintel Final		KITTI		Middlebury train		Middlebury test		Chairs	Time (sec)	
	train	test	train	test	train	test	AEE	AAE	AEE	AAE	test	CPU	GPU
EpicFlow	2.27	4.12	3.57	6.29	3.47	3.8	0.31	3.24	0.39	3.55	2.94	16	-
DeepFlow	3.19	5.38	4.40	0.21	4.58	5.8	0.21	3.04	0.42	4.22	3.53	17	-
EPPM	-	6.49	-	8.38	-	9.2	-	-	0.33	3.36	-	-	0.2
LDOF	4.19	7.56	6.28	9.12	13.73	12.4	0.45	4.97	0.56	4.55	3.47	65	2.5
FlowNetS	4.50	7.42	5.45	8.43	8.26	-	1.09	13.28	-	-	2.71	-	0.08
FlowNetS+v	3.66	6.45	4.76	7.67	6.50	-	0.33	3.87	-	-	2.86	-	1.05
FlowNetS+ft	(3.66)	6.96	(4.44)	7.76	7.52	9.1	0.98	15.20	-	-	3.04	-	0.08
FlowNetS+ft+v	(2.97)	6.16	(4.07)	7.22	6.07	7.6	0.32	3.84	0.47	4.58	3.03	-	1.05
FlowNetC	4.31	7.28	5.87	8.81	9.35	-	1.15	15.64	-	-	2.19	-	0.15
FlowNetC+v	3.57	6.27	5.25	8.01	7.45	-	0.34	3.92	-	-	2.61	-	1.12
FlowNetC+ft	(3.78)	6.85	(5.28)	8.51	8.79	-	0.93	12.33	-	-	2.27	-	0.15
FlowNetC+ft+v	(3.20)	6.08	(4.83)	7.88	7.31	-	0.33	3.81	0.50	4.52	2.67	-	1.12

Tabelle 2. EPE in pixels of state-of-the-art methods as well as FlowNets are given. Results of networks that were tested on the data they were trained on are indicated using parentheses [6].

be used as a partial solution to this.

KITTI. The output of the network is fairly good. Fine-tuning and variational refinement improves the results further. FlowNetS is better that FlowNetC in this case.

Flying Chairs. Since the networks are trained on this dataset, they perform best. 640 images were kept apart for validation as mentioned before. FlowNetC outperforms FlowNetS and is even better than the other state-of-the-art methods (Figure 13). Variational refinement however, seems to degrade the performance which is indicative of the fact that if the networks are provided a realistic training set, they might still perform better.

6.3 Analysis

Training data The networks were trained on Sintel exclusively, leaving aside a validation set and could predict the optical flow relatively well, along with data augmentation. There was 1 pixel increase in EPE in this method when compared to EPE of networks trained on Flying Chairs and fine-tuned on Sintel.

Training the networks on Flying Chairs without data augmentation resulted in an increased EPE of 2 pixels when tested on Sintel.

Comparing the architectures FlowNetS generalizes better to Sintel Final then FlowNetC although FlowNetC performs better on Flying Chairs and Sintel Clean. This shows that FlowNetC slightly overfits to the training data, proving that it adapts to the data presented. If more realistic training data were available, this could become an advantage.

When test data has large displacements, FlowNetC has problems. The results from KITTI show this. This is probably because the maximum displacement is limited as previously mentioned. Increasing the range can lead to lower computational efficiency.

7 Conclusion

Despite unrealistic training data, the networks could predict optical flow from input images well. The Flying Chairs dataset with rendered artificial images of transformed chairs was enough to predict optical flow in natural scenes. Thus proving the generalization capability of the network. FlowNet could also perform better than DeepFlow and EpicFlow. They might perform even better when realistic training data is available.

Literatur

[1] M. Aubry, D. Maturana, A. A. Efros, B. Russell, and J. Sivic. Seeing 3d chairs: exemplar part-based 2d-3d alignment using a large dataset of cad models. 2014.

[2] S. Baker, D. Scharstein, J.P.Lewis, S. Roth, M. J. Black, and R. Szeliski. A database and evaluation methodology for optical flow. *Technical Report MSR-TR-2009-179*, December 2009.

[3] L. Bao, Q. Yang, and H. Jin. Fast edge-preserving patchmatch for large displacment optical flow. *CVPR*, 2014.

[4] T. Brox and J. Malik. Large displacement optical flow: Descriptor matching in variational motion estimation. *PAMI*, 33(3):500–513, 2011.

[5] D. J. Butler, J. Wulff, G. B. Stanley, and M. J. Black. A naturalistic open source movie for optical flow evaluation. *A. Fitzgibbon et al. (Eds.)*, pages 611–625, October 2012.

[6] A. Dosovitskiy, P. Fischer, E. Ilg, P. Hausser, C. Hazirbas, V. Golkov, P. van der Smagt, D. Cremers, and T. Brox. Flownet: Learning optical flow with convolutional networks. *IEEE Int. Conference on Computer Vision (ICCV)*, 2015.

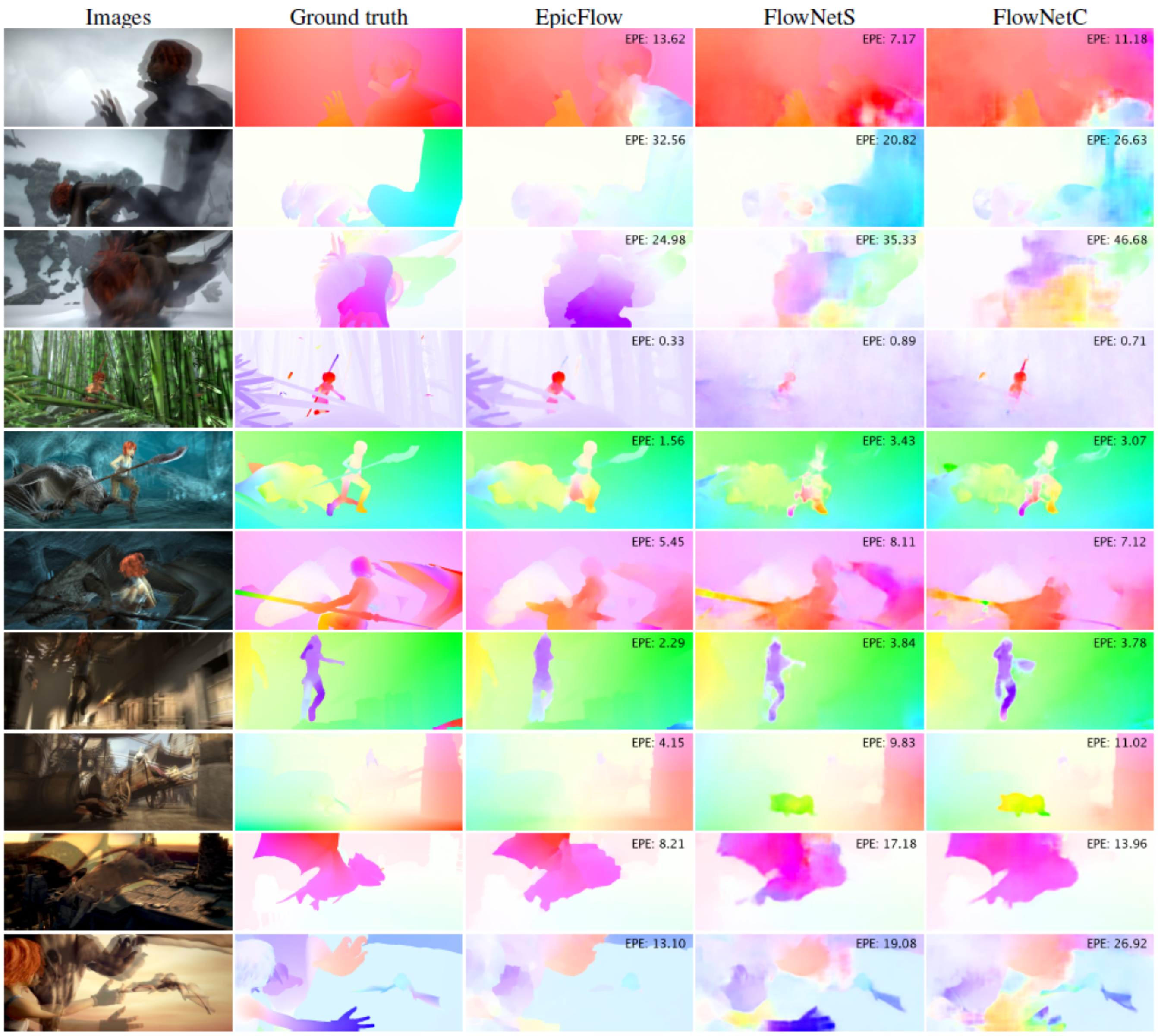


Fig. 12. Samples of optical flow prediction on the Sintel dataset.Predictions from EpicFlow, FlowNetS and FlowNetC are shown along with overlaid image pair and ground truth flow. The FlowNets perform better in terms of preserving finer details inspite of a higher EPE when compared to EpicFlow [6].

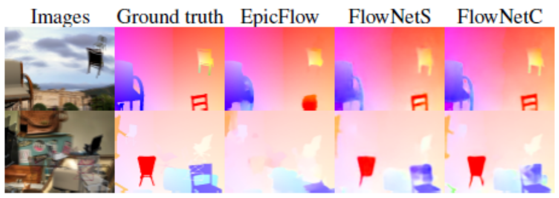


Fig. 13. Examples of optical flow estimation on samples of Flying Chairs dataset. The FlowNets predict the flow much better than EpicFlow [6].

[7] A. Dosovitskiy, J. T. Springenberg, M. Tatarchenko, and T. Brox. Learning to generate chairs, tables and cars with convolutional networks. *CV-PR*, 2015.

[8] C. en Lin. Introduction to motion estimation with optical flow. Nanonets, 2019.

[9] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun. Vision meets robotics: The kitti dataset. *International Journal of Robotics Research (IJRR)*, 2013.

[10] B. K. Horn and B. G. Schunck. Determining optical flow. *Artifical Intelligence*, 17:185–203, 1981.

[11] K. Konda and R. Memisevic. Unsupervised learning of depth and motion. *CoRR*, 2013.

[12] M. Leordeanu, R. Sukthankar, and C. Sminchisescu. Efficient closed-form solution to generalized boundary detection. *Proceedings of the 12th European Conference on Computer Vision - Volume Part IV*, pages 516–529, 2012.

[13] J. Long, E. Shelhamer, and T. Darrell. Fully convolutional networks for semantic segmentation. *CVPR*, 2015.

[14] D. P.Kingma and J. Ba. Adam: A method for stochastic optimization. *ICLR*, 2015.

[15] J. Revaud, P. Weinzaepfel, Z. Harchaui, and C. Schmid. Epicflow: Edge-preserving interpolation of correspondences for optical flow. *CVPR*, 2015.

[16] D. Sun, S. Roth, J. Lewis, and M. J. Black. Learning optical flow. *ECCV*, 2008.

[17] ujjwalkarn. An intuitive explanation of convolutional neural networks. The data science blog, August 2016.

[18] P. Weinzaepfel, J. Revaud, Z. Harchaoui, and C. Schmid. Deep flow: Large displacement optical flow with deep matching. *ICCV*, December 2013.