DispNet-CSS: Robust Vision Submission

Tonmoy Saikia, Eddy Ilg, Thomas Brox

June 4, 2018

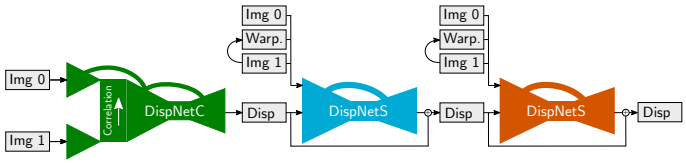

Figure 1: Overview of the DispNet-CSS architecture

1 Architecture

Motivated by the archictecture of FlowNet2 [2] we implement a stacked architecture for disparity named DispNet-CSS. This architecture is illustrated in Figure 1. We extend the original FlowNet2 architecture by modifying the connections in-between network stacks to be residual, to explicitly model refinement [4]. Furthermore, we make the following modifications in the architecture blocks:

1. scaling prediction values internally and
2. changing the level at which the correlation is performed.

1.1 Scaling of predictions

For training CNNs it is common to normalize input and output data to a common range. Let $\mathbf{f}$ be the output of the network, $\mathbf{y}^{\text{gt}}$ the ground truth and $\mathbf{y}$ our prediction. The original FlowNet [1, 2] provided the following implementation:

$$\min \mathcal{L}(\frac{1}{20} \cdot \mathbf{y}^{\text{gt}}, \mathbf{f}) \,,$$

$$\mathbf{y} = 20 \cdot \mathbf{f} \,,$$

Correlation level	Pred. scaling	EPE (Sintel)
2	No	5.62
3	No	3.33
3	Yes	3.19

Table 1: Impact of correlation depth (the level indicates after which convolution the correlation is defined in the network) and prediction scaling in DispNetC. Reported errors are from the Sintel train clean dataset. We observe a significant performance improvement by just changing the position of the correlation layer to be after the third convolution. Pred. scaling does not affect the EPE much, but results in less noisy predictions.

where $\mathcal{L}$ is the loss function. In other words, the ground truth was scaled down by a factor of 20. This leads to very small values in the network. We instead propose to scale up the values inside the network by removing the coefficient:

$$\min \mathcal{L}(\mathbf{y}^{\mathrm{gt}}, \mathbf{f}) \,,$$

$$\mathbf{y} = \mathbf{f} \,.$$

Prediction scaling doesn't significantly affect the performance quantitatively (see Table 1) but results in much less noisy disparity maps.

1.2 Depth of correlation layer

Another change is the level of the correlation layer. We move the correlation layer in DispNetC [3] one level up - from after conv2 to after conv3. By this change we increase the receptive field and obtain much better EPEs on Sintel (see Table 1). This is due to improved performance in large displacement regions, which are generally common in the disparity setting.

2 Training and evaluation

In this section we discuss the training details of our architecture.

2.1 Generic

We use the FlyingThings3D dataset to first train a generic version of our network from scratch. Each network in the stack is trained for $600k$ iterations. Each successive network in the stack is trained by fixing the weights of the previous network, i.e, we do not train networks in the stack jointly. Parameters for the first two networks are randomly initialized. However, due to the similar setting, the last network is initialized with parameters from the second network (which we call weight copying). The performance improvements can be observed in Table 2.

Configuration	Residual refinement	Weight copy	EPE
C	No	No	3.194
CS	No	No	2.634
CS	Yes	No	2.494
CSS	Yes	No	2.476
CSS	Yes	Yes	2.361

Table 2: Ablation study for training details of a single network and network stacks. Reported errors are from the Sintel train clean dataset. We train each network with the schedule of $600k$ iterations on FlyingThings3D [3]. Using residual connections [4] improves results. By transferring weights from the second to the third network (weight copy), we are able to obtain a further small improvement.

2.2 Fine-tuning

To specialize our generic network for the robust vision challenge we fine-tune our networks on a mixture of datasets: KITTI, Middlebury and ETH3D. To measure our performance we split each training set into train and validaton sets. Instead of re-sampling each dataset to a common resolution, we take the min width and height per dataset to generate 5 crops per sample. The crops are taken from the four corners and center. For training we use a random cropping with a crop size of $(768, 320)$. If an input sample is smaller than the crop size we upscale it by fixed factor. The results after finetuning each network in the stack is shown in Table 3. The first network is trained for $300k$ and the refinement networks are trained for $240k$.

Stack	ETH	MBH	KITTI15
C	0.36	2.36	1.11
CS	0.20	1.79	0.91
CSS	0.19	1.69	0.84

Table 3: Results of fine-tuning each network in the stack. The last row shows the results of the generic version for each dataset

2.3 Image scaling

For evaluating the test images on each dataset we scale the images by fixed factor. We observed this can help improve results slightly. We believe this is due the fact the correlation range is not optimally tuned for the input images. By changing the image resolution we affect the disparity range and thus the effective correlation between features. Based on the plot shown in Table 2, we scale all images from each dataset by factor of 1.4.

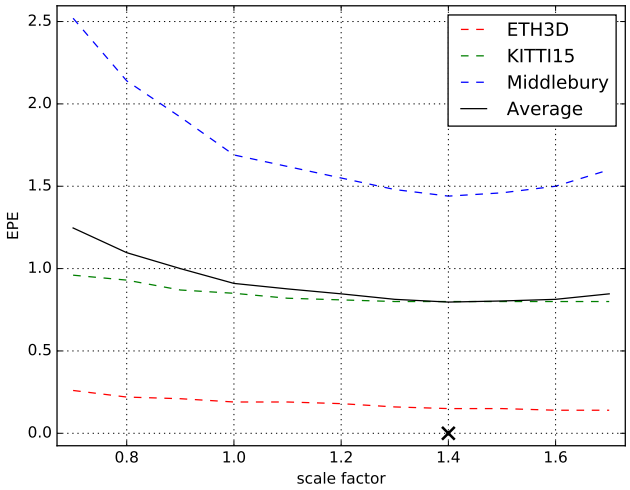

Figure 2: This plot shows the effect of image scaling (x-axis) on the validation errors (y-axis) for each dataset. We observe that up-scaling images slightly helps improve the validation errors. Based on this plot we choose 1.4 to upscale each test image.

References

[1] A. Dosovitskiy, P. Fischer, E. Ilg, P. Häusser, C. Hazırbaş, V. Golkov, P. v.d. Smagt, D. Cremers, and T. Brox. Flownet: Learning optical flow with convolutional networks. 2015.

[2] E. Ilg, N. Mayer, T. Saikia, M. Keuper, A. Dosovitskiy, and T. Brox. Flownet 2.0: Evolution of optical flow estimation with deep networks. 2017.

[3] 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. 2016.

[4] Jiahao Pang, Wenxiu Sun, Jimmy S. J. Ren, Chengxi Yang, and Qiong Yan. Cascade residual learning: A two-stage convolutional neural network for stereo matching. 2017.