Supplementary material of: CNN-based Patch Matching for Optical Flow with Thresholded Hinge Embedding Loss

Probability for distribution	Distribution
2/7	$N_{10}(p_2)$
1/7	$N_{20}(p_2)$
1/7	$N_{50}(p_2)$
1/7	$N_{100}(p_2)$
1/7	$N_{200}(p_2)$
1/7	$N_{\infty}(p_2)$

Table 1. Distribution of negative training samples $N(p_2)$.

1. Introduction

In this supplementary material we present additional details and results for our main paper. In Section 2 we present the distribution of our negative training samples $N(p_2^+)$ which we introduced in Section 3.2 in the paper. Then, we show the principle of our test time speedup in Section 3. In Section 4 we introduce the faster CNN structure mention in Section 4.3 in the paper. In Figure 3 in the paper we omitted one plot for clarity reasons. The full figure is shown in Figure 2. The corresponding section is Section 4. Finally, in Section 5 we present some additional plots. These are based directly on robustness r instead of $E(net1, net2)$. While, $E(net1, net2)$ as we used it in the paper makes it easier to compare networks it omits the absolute component. We also plot the average L_2 distance of different spatial distances in Figure 5. Figure 4 shows that we report most failures for small distances as argued in Section 4.2 in the paper.

2. Distribution of negative training samples

Here we introduce the distribution $N(p_2^+)$ which we used for choosing the negative training samples. With $N(p_2^+)$ we wanted to create distribution that should strongly prefer closer (=harder to train) samples, but still not ignore far away samples. Our basic idea was something like $P(N_{idea}(p_2^+) = p_2) = 1/max(10, ||p_2^+ - p_2||^2)$/#norm. (#norm = normalization factor). However this is difficult to implement. Instead the distribution $N(p_2^+)$ is defined as meta distribution that samples from distributions $N_x(p_2^+)$ with a certain probability. The distributions it samples from and the respective probabilities are shown in Table 2. A distribution $N_x(p_2^+)$ is an uniform distribution that contains all samples that are at least 2 pixels and at most x pixels away from p_2^+ in image space. Note that for instance $N_{1300}(p_2^+)$ is identical to $N_{\infty}(p_2^+)$ due to the limited image size. $N(p_2^+)$ approximates $N_{idea}(p_2^+)$ (with little more focus to closer samples) and is easy to implement with basic random number generators as it consists only of uniform distributions.

3. Test time speedup trough weight sharing

Our CNNs are trained on single patches, but in the test phase used to calculate feature vectors for patches around each pixel in an image (in a sliding window manner). As illustrated in Figure 1 (please read its caption) CNNs of patches of neighboring pixels share many intermediate layer convolution results. To avoid redundant recalculations, convolutions of a layer can simply be performed on the whole image at once instead of single patches. For poling layers (or other layers with stride) the layer has to be calculated several times with different pooling offsets. This is also demonstrated in Figure 1. In the illustration two pooling layers are required (illustrated by the green and blue color). For a 2 dimensional image and our 2x2 pooling, 2*2 = 4 poolings are required (pool(x,y), pool(x+1,y), pool(x,y+1) and pool(x+1,y+1)). Furthermore, in case of two pooling layers in the network every pooling layer of the first pooling layer must be pooled again four times in the second pooling layer which leads all together to (2*2)*(2*2) = 16 poolings in the 2. layer. In our implementation we treat the 4 poolings of the first pooling layer and the 16 poolings on the second pooling layer as individual samples for the subsequent layers. In a final stage we recompose the original image from the 16 mini images.

Layer	1	2	3	4	5	6	7	8
Type	Conv	MaxPool	Conv	Conv	MaxPool	Conv	Conv	Conv
Input size	56x56	52x52	26x26	22x22	18x18	9x9	5x5	1x1
Kernel size	5x5	2x2	5x5	5x5	2x2	5x5	5x5	1x1
Out. channels	64	64	80	128	160	196	256	256
Stride	1	2	1	1	2	1	1	1
Nonlinearity	Tanh	-	Tanh	Tanh	-	Tanh	Tanh	Tanh

Table 2. The improved CNN architecture mentioned in the paper (post experiment).

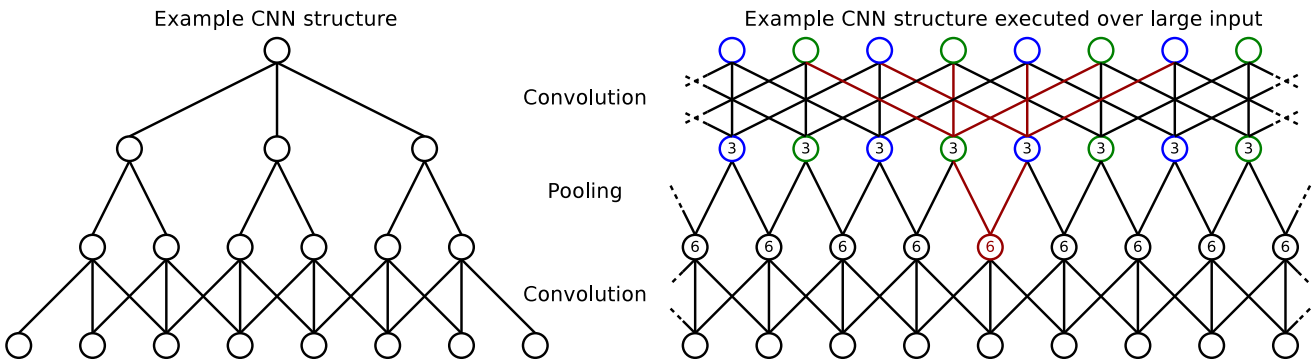

Figure 1. Left: A simplified 2 dimensional CNN structure for patch feature creation that contains one pooling layer and two convolution layers. Right: if this CNN is executed at each pixel position of an image to create features for every position many intermediate layer results are shared between networks. The numbers in the nodes state how often a node is shared. The red connections show how the red node is shared. Pooling with stride two halves the output resolution. Thus, we need two pooling layers: the original one (blue) and one shifted by one pixel (green) to avoid halving the output resolution. Outputs always only depend on e one kind of pooling.

4. Faster CNN architecture

The improved CNN architecture mentioned in the paper that only requires 2.5s instead of 4.5s (for all CNN operations needed for an image pair) is shown in Figure 2. For this architecture we reduced the number of channels in the upper layers, as these are most costly when calculating features for each pixel with weight sharing (see last section). Still, on our validation set this architecture even performed slightly better (but not mentionable better). Note that for our paper we did not spend much effort in improving the runtime of our CNN architecture, as our (so far still CPU based) patch matching requires more time than CNN feature creation.

5. Training a CNN with the distribution of a downsampled CNN

Instead of training a CNN on a 2x down-sampled image we can also just train a CNN on the highest resolution with the distribution $N^{\times 2}(p_2^+) = 2N(p_2^+) - p_2^+$ of the 2x downsampled image i.e. distances between p_2^+ and p_2^- in $N(p_2)$ are multiplied by 2. Figure 2 contains this result additionally to the results already presented in the paper. As can be seen in the figure, this is not a good solution compared to real downsampling.

6. Additional Plots

Figure 3 shows the one minus matching robustness ($1 - r$) curves for different losses and SIFTFlow regarding optical flow displacement.

Figure 4 shows the matching robustness curves r regarding distances between the correct match p_2^+ and a wrong match p_2^-. The robustness is much smaller for small distances than for large ones. This shows that training of small distances is much more challenging. Still, conventional CNN patch learning papers that treat patch matching on an abstract level usually do not consider these "harder" cases.

Note that despite the much higher robustness, robustness differences for larger distances are also very relevant, as there are more points with large distances than with small distances (more potential failure cases). Furthermore, an outlier with a larger distance causes a larger endpoint error. Also, with patch matching a successful match of large distance is often a mandatory requirement to be even able to perform a smaller distance match.

Figure 5 shows the average L_2 distance for different distances between the correct match p_2^+ and a wrong match p_2^-. With larger t these curves become flatter for L_t.

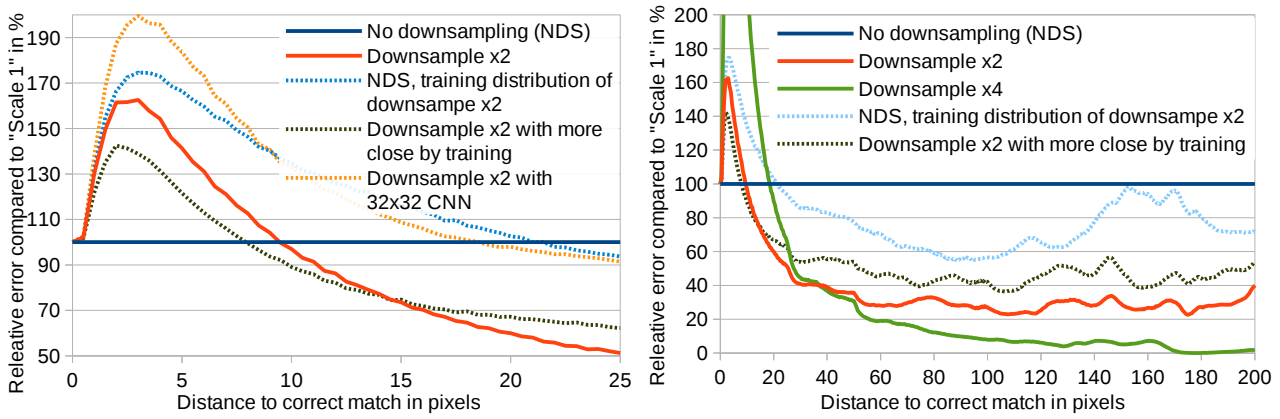

Figure 2. Features created on lower scales are more accurate for large distances but less accurate for small distances. Here we also include the curve of a CNN that was on the highest resolution trained with the distribution of a 2x downsampled resolution.

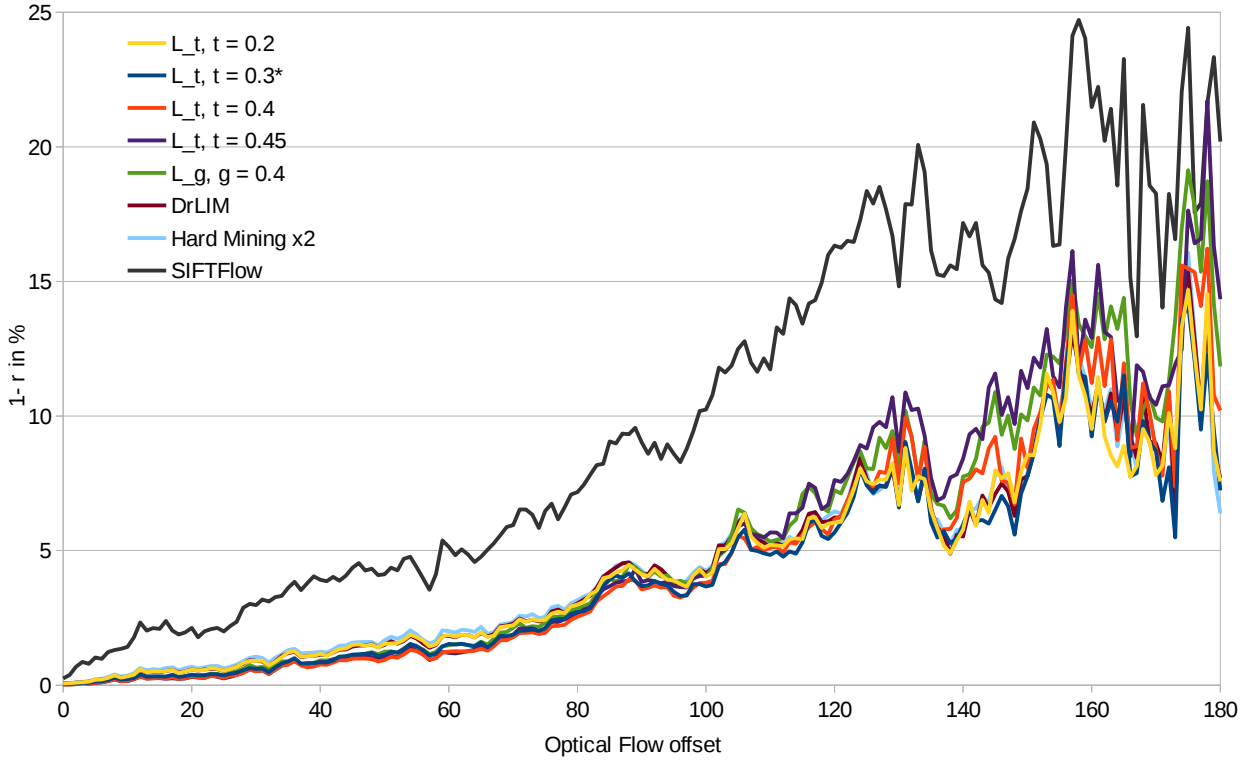

Figure 3. Absolute robustness plots for different flow displacements (We use 1-r not r for this figure).

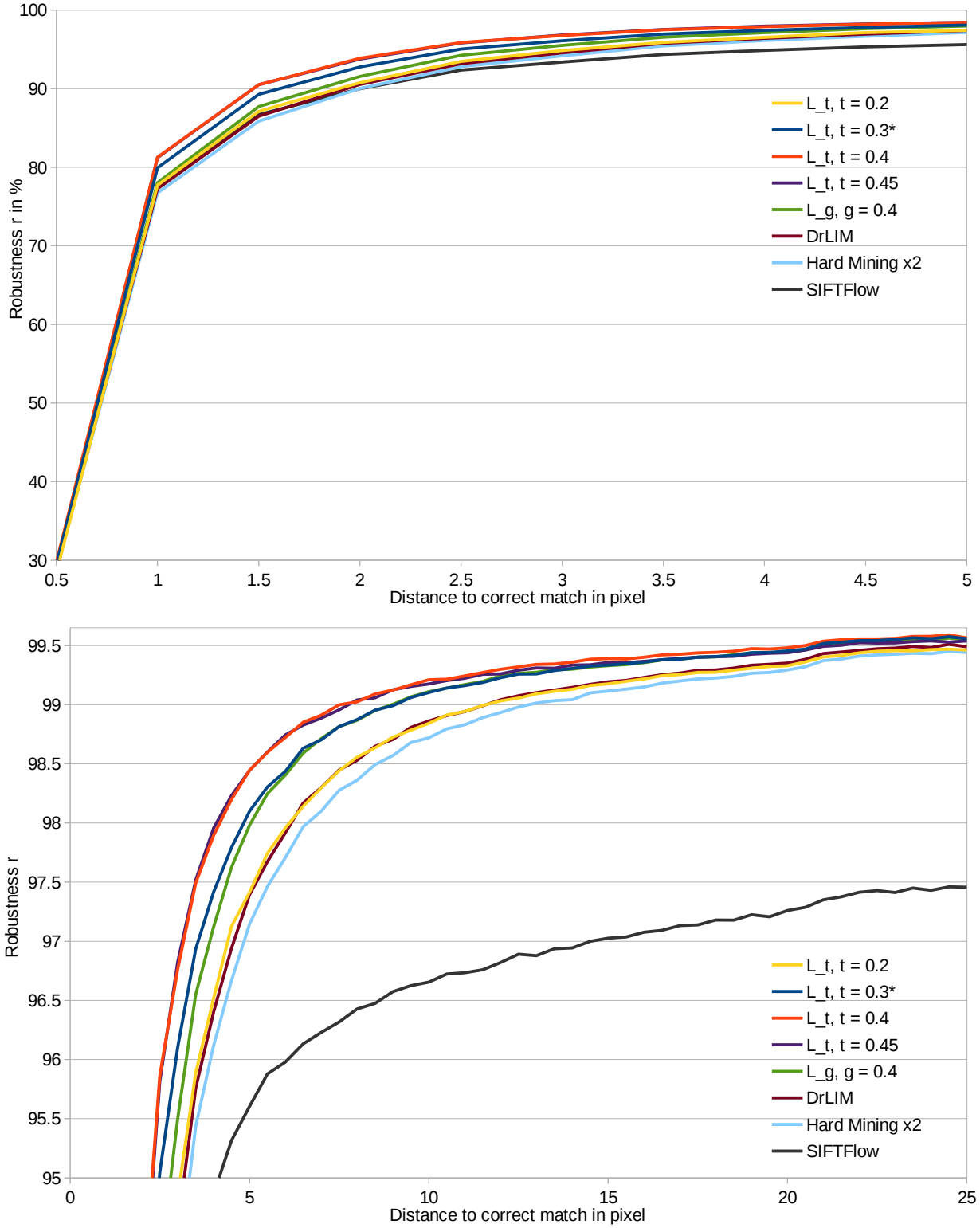

Figure 4. Absolute robustness plots for different distances between the correct match p_2^+ and a wrong match p_2^-.

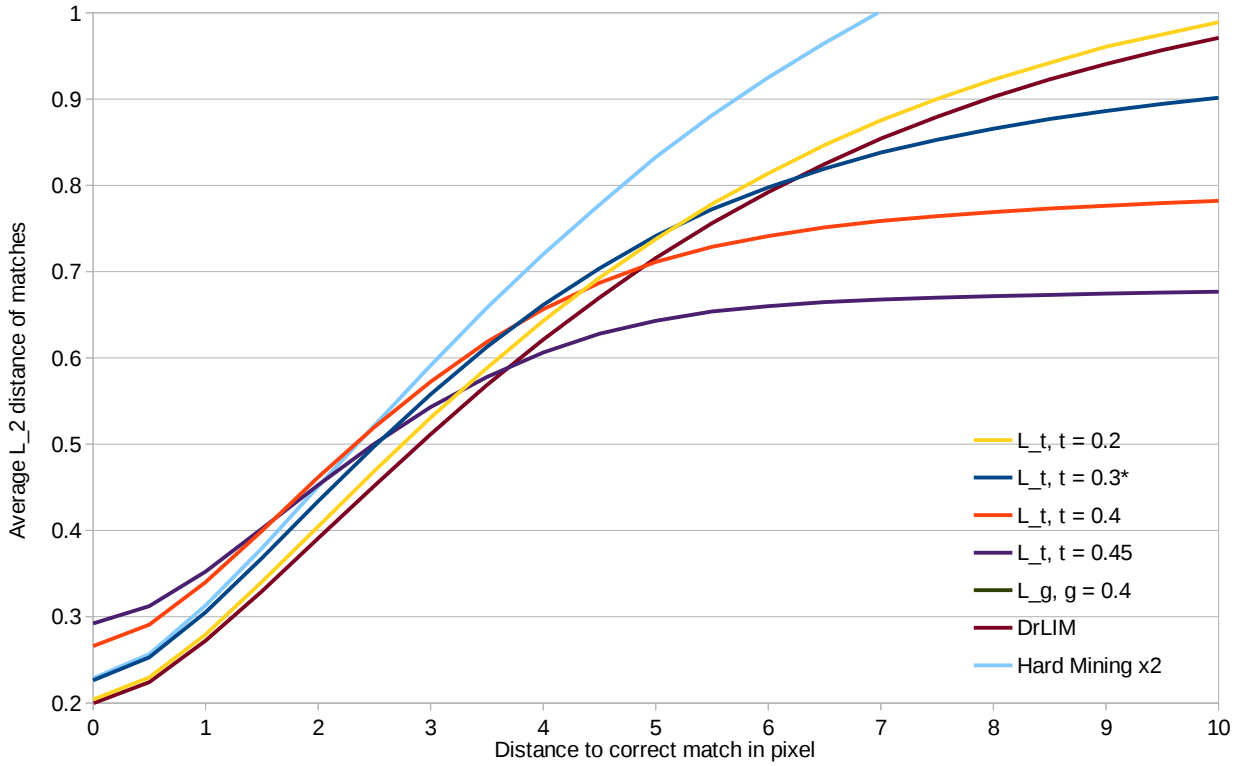

Figure 5. Average L_2 distance for different distances to the correct match for different losses.

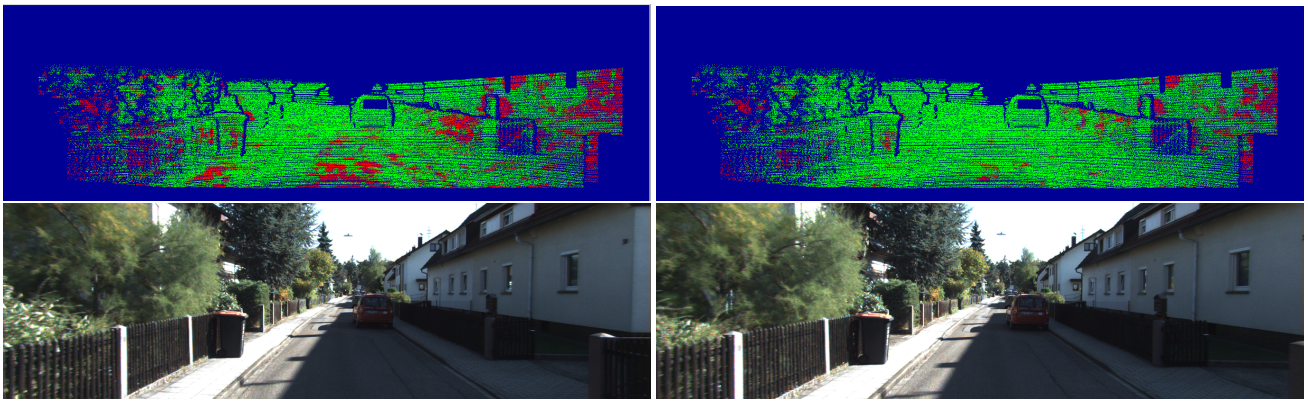

Figure 6. Pixelwise robustness for the first image in the validation set. Blue: No ground truth. Red: Less than 95% matching robustness. Green: 100% robustness. Color between green an red: matching robustness between 95% and 100%. On the top left is the result for the hinge loss L_h. On the top right for $L_t, t = 0.3$. On the bottom are the corresponding views. Most errors (low robustnesses) accumulate in large connected regions. However, with our loss $L_t, t = 0.3$ there are less and smaller error regions.