Descriptor Matching with Convolutional Neural Networks: a Comparison to SIFT

Philipp Fischer[*][†]
Department of Computer Science
University of Freiburg
fischer@cs.uni-freiburg.de

Alexey Dosovitskiy[†]
Department of Computer Science
University of Freiburg
dosovits@cs.uni-freiburg.de

Thomas Brox
Department of Computer Science
University of Freiburg
brox@cs.uni-freiburg.de

Abstract

Latest results indicate that features learned via convolutional neural networks outperform previous descriptors on classification tasks by a large margin. It has been shown that these networks still work well when they are applied to datasets or recognition tasks different from those they were trained on. However, descriptors like SIFT are not only used in recognition but also for many correspondence problems that rely on descriptor matching. In this paper we compare features from various layers of convolutional neural nets to standard SIFT descriptors. We consider a network that was trained on ImageNet and another one that was trained without supervision. Surprisingly, convolutional neural networks clearly outperform SIFT on descriptor matching.

1 Introduction

For years, local descriptors based on orientation histograms, such as SIFT, HOG, SURF [1, 3, 12], etc. have dominated all domains of computer vision. The considerable progress we have seen in recognition is largely due to them: image retrieval is based on aggregated SIFT descriptors, structure from motion started to work reliably due to SIFT correspondences, and even in low level problems, such as optical flow estimation, orientation histograms help deal with large motion [2].

For the last few years, SIFT has been challenged by attempts to learn features automatically from datasets. The convolutional neural network architecture by Krizhevsky et al. [9] has finally been successful in outperforming handcrafted features in basically all important recognition tasks. Although it is not fully clear yet whether the network benefits mostly from the massive amount of training data in a learning-by-heart fashion or if it is indeed able to learn abstract visual concepts that generalize to unseen data, there is indication that the latter is at least partially true. There are several papers demonstrating that a network trained for classification on ImageNet also performs very well on other datasets and recognition tasks [4, 6, 16, 17].

From the beginning, SIFT was not only successful in recognition but maybe even more so in descriptor matching. Still, there is not any study yet about the performance of convolutional networks for matching, though there are many computer vision tasks outside recognition that rely on descriptor

[*]Supported by the Deutsche Telekom Stiftung
[†]Both authors contributed equally

matching. In this paper we aim to find out how neural networks perform on matching tasks, as compared to SIFT. When we started this study, our expectations were low: features from a convolutional neural network have been optimized to distinguish object classes. How should this help in a generic matching task? We expected that the specialization to discriminate classes would lead to ignorance of subtle structures that are important for matching. In fact, we were interested in how this effect hits the various network layers. Is there a network layer that still provides reasonable descriptors for a matching task?

We made our comparison more exciting by including a convolutional neural network that has been trained without object class annotation. It goes back to a very recent idea by Dosovitskiy et al. [5] and approaches two limitations of a supervised ConvNet: (1) no labeled data is needed for training the network and (2) the training objective may fit better to a descriptor matching task.

We compared the descriptors on the descriptor matching benchmark by Mikolajczyk et al. [15], which is the most popular benchmark dataset for this task. The size of this benchmark is very small for today's standards and statistical significance of results can hardly be judged. For this reason, we created an additional larger dataset to double check the results we obtained on the standard benchmark. The evaluation yields a clear and surprising result: descriptors extracted from convolutional neural networks perform consistently better than SIFT *also* in the low-level task of descriptor matching. Another interesting finding is that the unsupervised network slightly outperforms the supervised one.

2 Feature Learning with Convolutional Neural Nets

In the past, convolutional neural networks (CNNs) were used mainly for digit and document recognition [10, 11] as well as small-scale object recognition [8]. Typical training datasets were MNIST and CIFAR-10, each including $60,000$ images of roughly 30×30 pixels. Thanks to an optimized GPU implementation and new regularization techniques, Krizhevsky et al. [9] successfully applied CNNs to classification on the ImageNet dataset with over 1 million labeled high resolution images. The feature representation learned by this network shows excellent performance not only on the ImageNet classification task it was trained for, but also on a variety of other recognition tasks [4, 6, 16, 17].

A typical criticism of using huge labeled datasets for training CNNs is that obtaining such data requires expensive manual annotation[1]. To address this drawback of traditional supervised training, Dosovitskiy et al. [5] proposed an unsupervised approach to train CNNs by making use of data augmentation. We included this algorithm in our comparison to analyze how much the performance of the supervised network benefits from the given high-level information. Class labels may not be as valuable in a descriptor matching task as in a classification task; they might even be disadvantageous.

For training CNNs and subsequent feature extraction we use the *caffe* code [7].

2.1 Supervised Training

Instead of training a network on ImageNet from scratch, we used a pre-trained model available at [7]. The architecture of the network follows Krizhevsky et al. [9]: the network contains 5 convolutional layers followed by 2 fully connected layers and a softmax classifier on top. The sizes of convolutional filters are, respectively, 11×11, 5×5, 3×3, 3×3, 3×3 pixels. The numbers of filters are 96, 256, 384, 384, 256, respectively. Each fully connected layer contains 4096 neurons. Max-pooling and local response normalization are performed after some layers. The activation function is a rectified linear unit (ReLU) nonlinearity, dropout is applied in fully connected layers. We refer the reader to [9] for more details on the architecture and training algorithm.

2.2 Unsupervised Training

We performed unsupervised training of a CNN as described in [5]. The main idea of the approach is to create surrogate labels for an unlabeled set of training patches and thereby replace the labeled

[1]One could argue, of course, that as the labeled data are already available, there is no additional cost in using it. However, there is additional cost in scaling up the supervised algorithms and, hence, the labeled datasets.

data necessary to train the CNN. We briefly review this method and describe our specific design choices, the reader can find more details in the original paper [5].

In order to create the surrogate data, the algorithm requires an unlabeled dataset as input. Instead of STL-10 unlabeled dataset as in [5], we used random images from Flickr because we expect those to be better representatives of the distribution of natural images. Next $N = 16000$ 'seed' patches of size 64×64 pixels were extracted randomly from different images at various locations and scales. Each of these 'seed' patches was declared to represent a surrogate class of its own. These classes were augmented by applying $K = 150$ random transformations to each of the 'seed' patches. Each transformation was a composition of random elementary transformations. These included translation, scale variation, rotation, color variation, and contrast variation, as in the original paper, but also blur, which is often relevant for matching problems. As a result we obtained a surrogate labeled dataset with N classes containing K samples each. We used these data to train a convolutional neural network.

The surrogate dataset is much less diverse than the real-life ImageNet dataset: within each class there is no inter-instance variation or large 3D viewpoint change. To avoid overfitting, we used a smaller network than the one trained on ImageNet. It contains 3 convolutional layers, 1 fully connected layer and softmax classifier on top. Convolutional layers contain: 64 filters 7×7 pixels, 128 filters 5×5 pixels, 256 filters 5×5 pixels. The fully connected layer contains 512 neurons. The first convolutional layer is applied with a stride of 2 pixels. First two convolutional layers are followed by 2×2 max-pooling. No local response normalization is performed. Similarly to the ImageNet network, we used ReLUs and dropout.

3 Experimental Study

While recognition tasks such as image classification and object detection rely on the underlying semantic structure of the scene, we expect the matching of interest points to be independent of that information. This makes an evaluation of automatically learned features on matching tasks interesting and leads to the question whether a feature representation trained for classification can also perform well as a local interest point descriptor.

We compare the features learned by supervised and unsupervised convolutional networks, as well as two baselines: SIFT and raw RGB values. SIFT is the preferred descriptor in matching tasks (see [14] for a comparison), while raw RGB patches serve as a weak 'naive' baseline. The focus of our interest is matching performance, therefore we first computed regions of interest in all images (see the details below) and used these as the input to all description methods. This allows us to analyze specifically the performance of descriptors, not detectors.

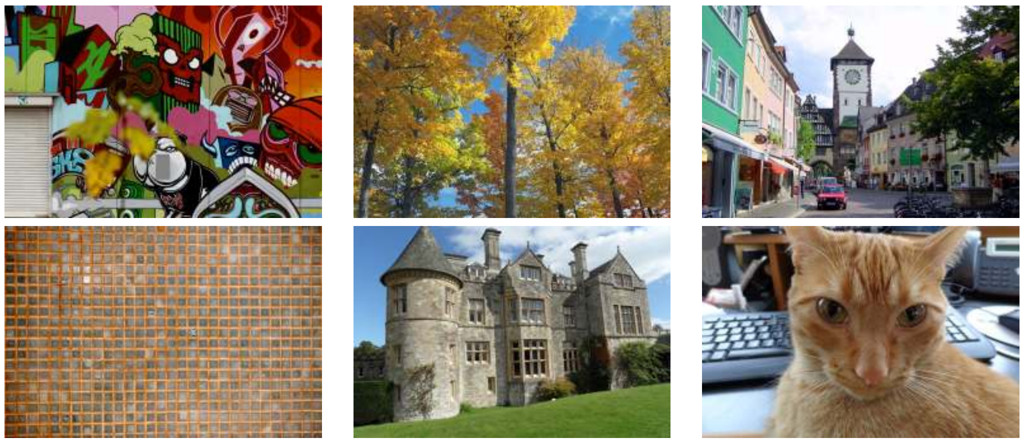
Figure 1: Some base images used for generating the dataset.

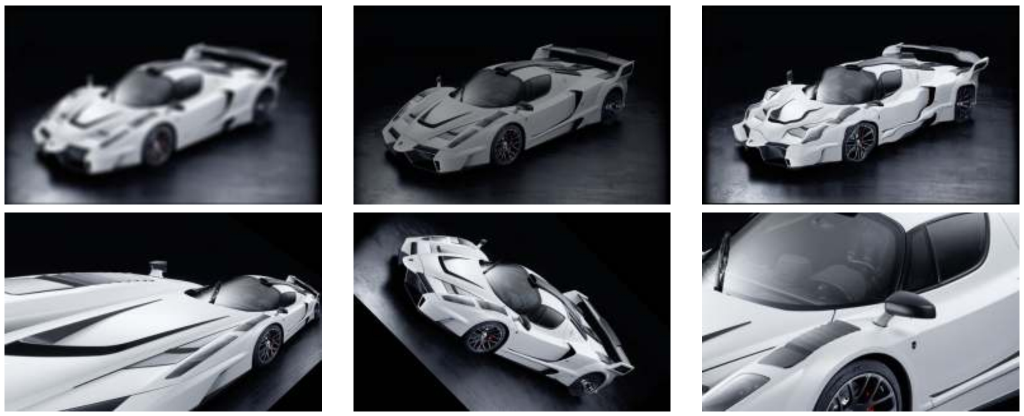
Figure 2: Most extreme versions of the transformations applied to the base images. From left to right: blur, lighting change, nonlinear deformation, perspective change, rotation, zoom.

3.1 Datasets

The common matching dataset by Mikolajczyk et al. [15] contains only 48 images. This dataset size limits the reliability of conclusions drawn from the results, especially as we compare various design choices, such as the depth of the network layer from which we draw the features. We set up an additional dataset that contains 416 images. It was generated by applying 6 different types of transformations with varying strengths to 16 base images we obtained from Flickr. These images were not contained in the set we used to train the unsupervised CNN. Figure 1 shows some of them.

To each base image we applied the geometric transformations *rotation*, *zoom*, *perspective*, and *nonlinear deformation*. These cover rigid and affine transformations as well as more complex ones. Furthermore we applied changes to *lighting* and focus by adding *blur*. Each transformation was applied in various magnitudes such that its effect on the performance could be analyzed in depth. The transformations are shown in Figure 2. For each of the 16 base images we matched all the transformed versions of the image to the original one, which resulted in 400 matching pairs.

The dataset from Mikolajczyk et al. [15] was not generated synthetically but contains photos taken from different viewpoints or with different camera settings. While this reflects reality better than a synthetic dataset, it also comes with a drawback: the transformations are directly coupled with the respective images. Hence, attributing performance changes to either different image contents or to the applied transformations becomes impossible. In contrast, the new dataset enables us to evaluate the effect of each type of transformation independently of the image content. As we shall see, the results on both datasets are consistent, which allows us to draw conclusions with a high level of certainty.

3.2 Performance Measure

To evaluate the matching performance for a pair of images, we strictly followed the procedure described in [14]. We first extracted elliptic regions of interest and corresponding image patches from both images using the *maximally stable extremal regions* (MSER) detector [13]. We chose this detector because it was shown to perform consistently well in [15] and it is widely used. Next we computed the descriptors of all extracted patches and greedily matched them based on the Euclidean distance. This yields a ranking of descriptor pairs. A pair is considered as a true positive if the ellipse of the descriptor in the target image and the ground truth ellipse in the target image have an intersection over union (IOU) of at least 0.6. All other pairs are considered as false positives. Assuming that a recall of 1 corresponds to the best achievable overall matching given the detections, we computed a precision-recall curve. The average precision, i.e., the area under this curve, was used as performance measure.

3.3 Parameter Choices for Descriptor Computation

The MSER detector returns ellipses of varying sizes, depending on the scale of the detected region. Descriptors are derived from these elliptic regions by normalizing the image patch to a fixed size. It is not immediately clear which patch size is best: larger patches provide a higher resolution, but enlarging them too much may introduce interpolation artifacts and the effect of high-frequency noise may be emphasized. Therefore, we optimized the patch size on our larger dataset for each method.

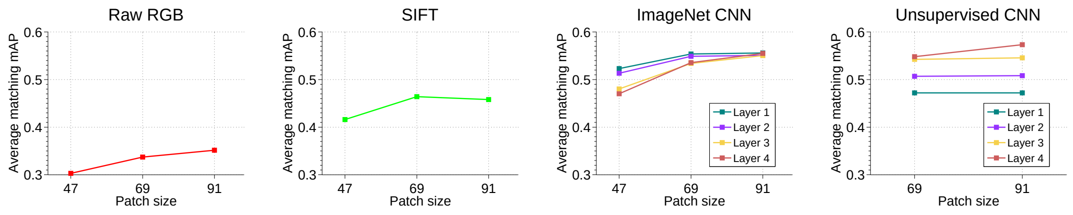

Figure 3: Parameter optimization on the larger dataset. For raw RGB and SIFT we optimized the patch size to which the detected region is normalized. For the neural nets we optimized the patch size and the network layer from which we collect the features.

Figure 3 shows the average performance of each method when varying the patch size. We did not test patch size 47 for the unsupervised network because it was trained on 64×64 images and, hence, cannot be applied to smaller images. While SIFT prefers a patch size of 69×69, the neural nets work better with larger patch sizes. As usual, SIFT divides the image patch into 4 by 4 cells. Hence, resizing may blur important gradients but would not bring in any new information. On the other hand, the neural networks have fixed filter and pooling sizes. Therefore, the size of the receptive field (region of the input image which affects a single neuron) is fixed, and adjusting the input patch size to best fit this receptive field size improves the performance. Higher layers benefit more from larger patch sizes as they have larger receptive fields.

A technical detail regarding feature extraction with neural nets is that because of differences in the architecture the unsupervised network produces larger feature maps than the supervised one. To compensate for this we applied additional 2×2 max-pooling to the outputs of the unsupervised network.

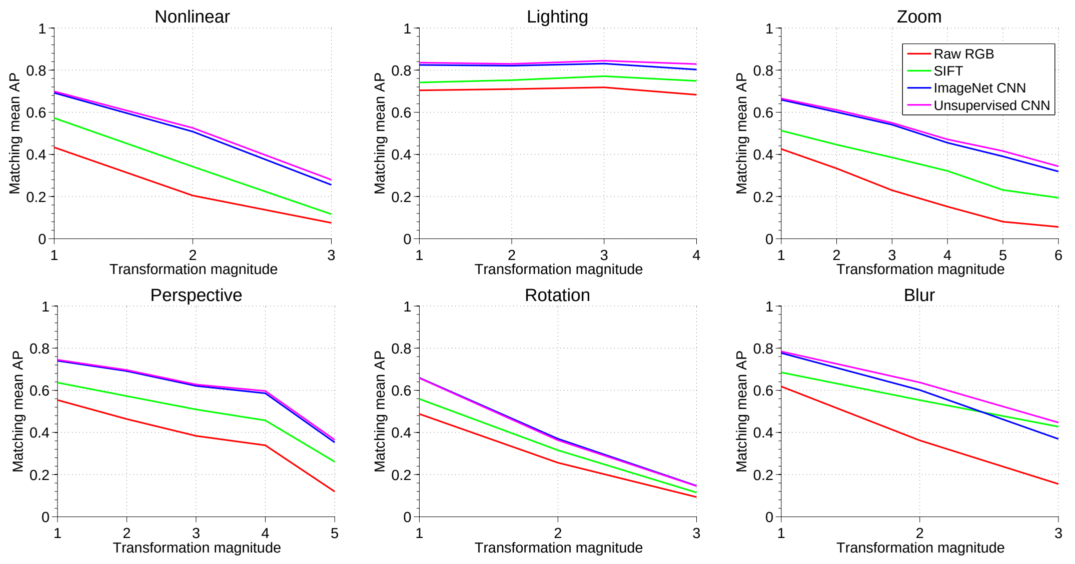

Figure 4: Mean average precision on the larger dataset for various transformations. Except for the blur transformation, both neural nets perform consistently better than SIFT. The unsupervised net is also better on blur.

When using convolutional neural networks for region description, aside from the patch size there is another fundamental choice – the network layer from which the features are extracted. While lower layers are closer to the data, higher layers provide more abstraction. We extracted features from various layers of neural networks and measured their performance. While the unsupervised CNN clearly prefers features from higher network layers, the ImageNet CNN does not have a preference if the optimal patch size is used. In the remaining experiments we used features from the 4th layer for both networks.

3.4 Results

Figure 4 shows the mean average precision on the various transformations of the new dataset using the optimized parameters. Surprisingly, both neural nets perform much better than SIFT on all transformations except blur. The unsupervised network is superior to SIFT also on blur, but not by a large margin. Interestingly, the difference in performance between the networks and SIFT is typically as large as between SIFT and raw RGB values. This shows the significance of these results: CNNs outperform one of the best hand-crafted descriptors as much as this descriptor outperforms the naive baseline.

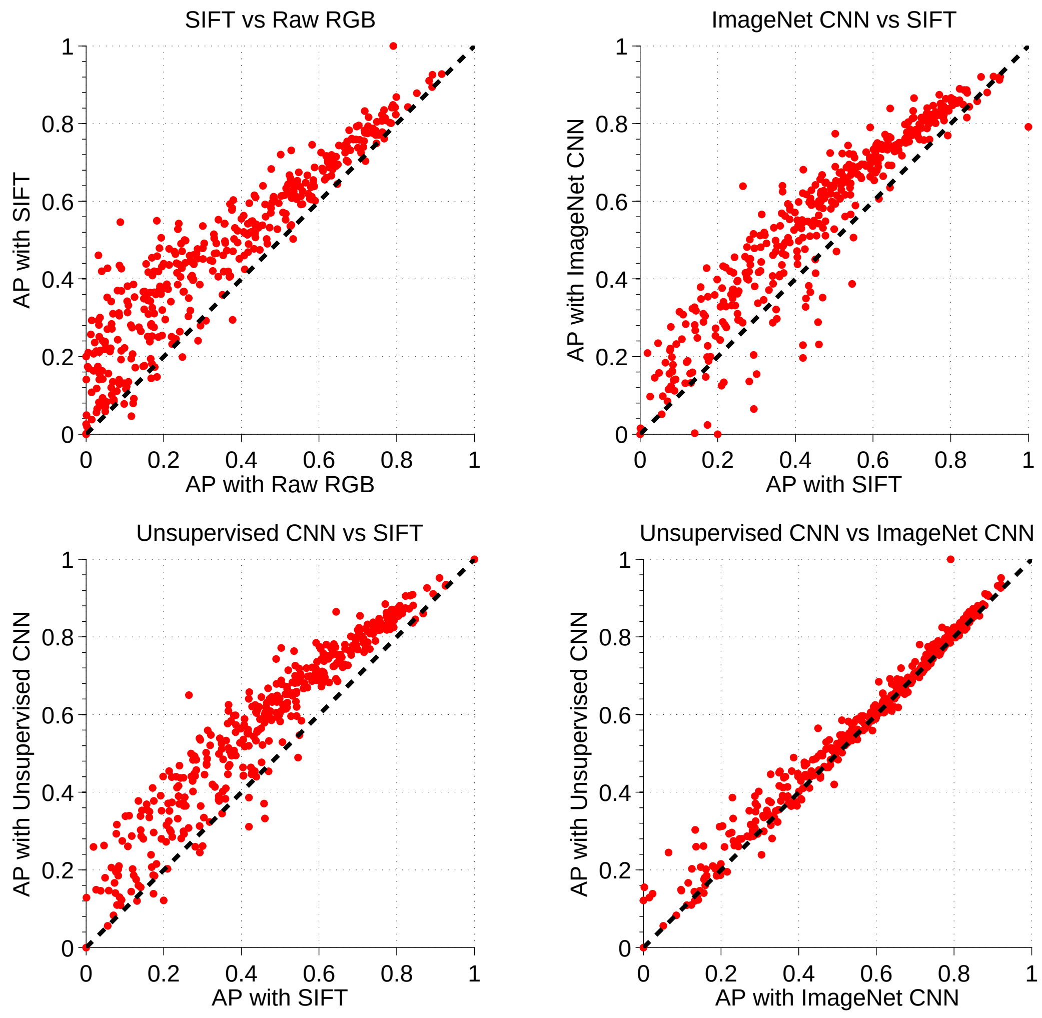

Figure 5: Scatter plots for different pairs of descriptors on the larger dataset. Each point in a scatter plot corresponds to one image pair, and its coordinates are the AP values obtained with the two descriptors being compared. The unsupervised CNN improves performance over SIFT as much as SIFT improves performance over raw RGB patches (see the two left images).

An exception is strong blur, which seems to be problematic especially for the supervised CNN. The unsupervised training explicitly considers blur, which is why the unsupervised CNN suffers less from it. Since blurred images can also appear in recognition tasks, making the CNN architecture more robust to this transformation could potentially improve classification performance. This especially applies to recognizing small objects: in the majority of algorithms they are scaled up to a fixed size, which results in blurring the image.

For an in-depth comparison of the methods, we plot the single AP values for each of the 400 image pairs in Figure 5. A point in one of the scatter plots corresponds to one image pair, while its coordinates are the AP values of the two methods being compared. Points above the diagonal indicate better performance of the first method and points below the diagonal show that the AP of the second method is higher.

The top-left graph compares the naive RGB descriptor to SIFT and shows how SIFT is clearly better in almost all image pairs except for a few. The same is true for the unsupervised CNN compared to SIFT in the lower left graph, while the supervised net (top right graph) exposes some more outliers, mainly due to its problems with blur. In the lower right we compare the two CNNs: in most cases the unsupervised net is better than the supervised one, yet the margin is smaller than in the other pairwise comparisons.

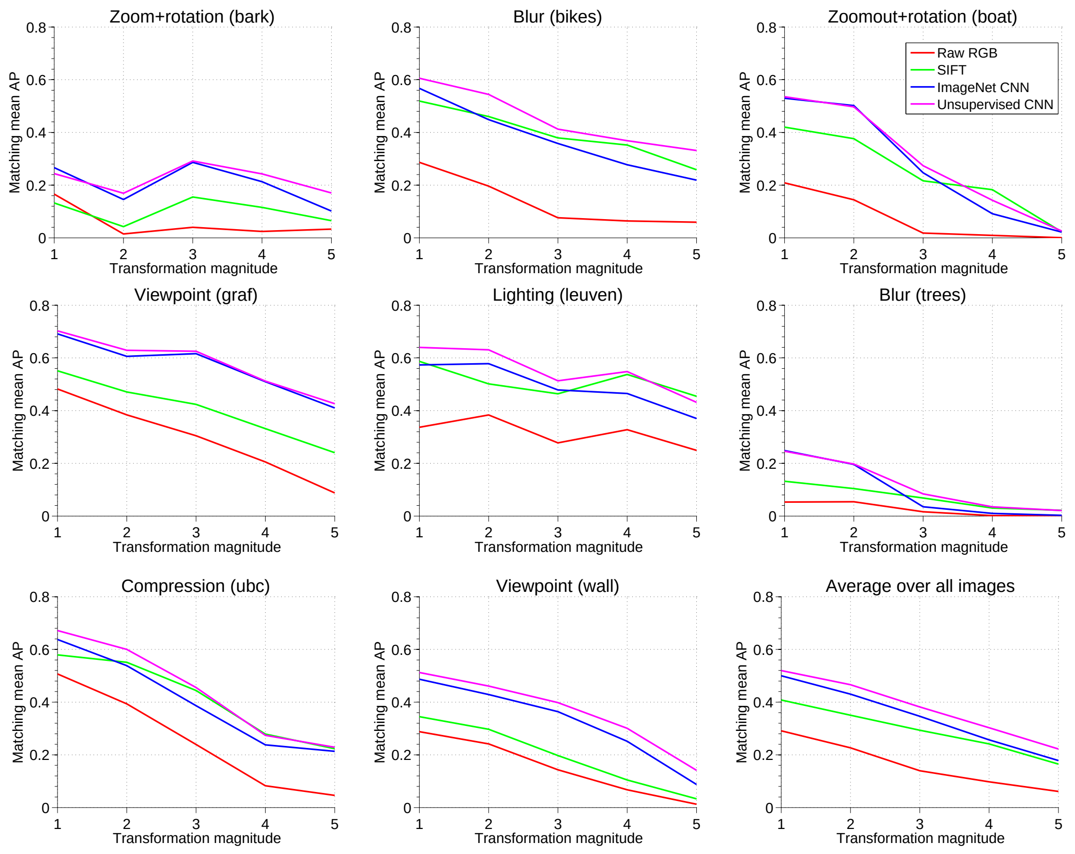

Figure 6: Mean average precision on the Mikolajczyk dataset. The bottom right plot shows the average over the whole dataset. On average both nets outperform SIFT. For some transformations (blur, compression) SIFT is better than ImageNet net, but not the unsupervised net.

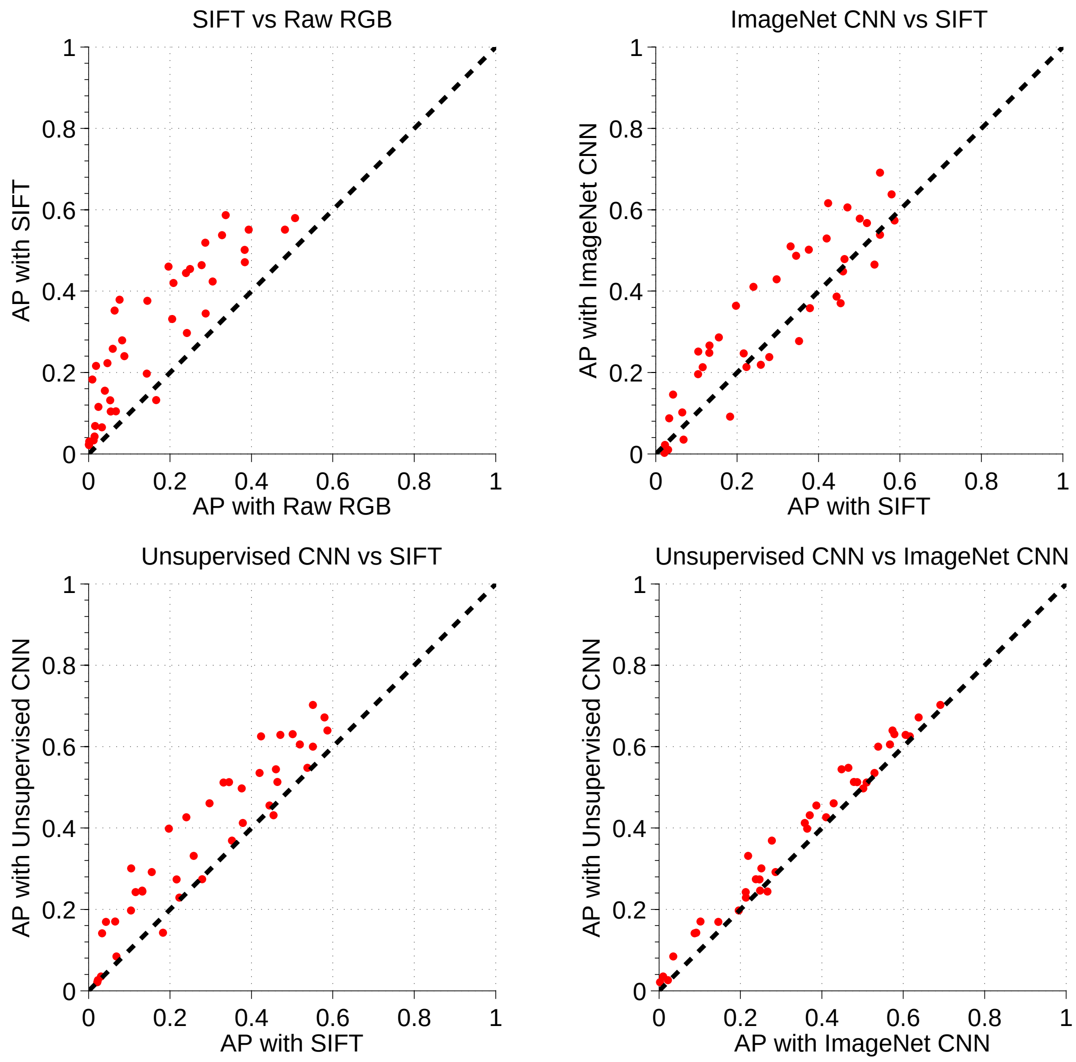

Figure 7: Scatter plots for different pairs of descriptors on the Mikolajczyk dataset. Each point in a scatter plot corresponds to one image pair, and its coordinates are the AP values obtained with the two descriptors being compared.

Finally, Figure 6 shows the mean average precision on the dataset from Mikolajczyk. The same effect as seen on the larger dataset can be observed also here, though less clearly because the dataset is much smaller. Again CNNs have problems with strong blur (and the related zoomout), but outperform SIFT on the other transformations. In general, the performance of CNNs is consistently better for small and medium transformations and has more problems with extreme transformations. In Figure 7, we provide scatter plots also for the Mikolajczyk dataset, which confirm the findings from Figures 5 and 6.

Computation times per image are shown in Table 1. SIFT computation is clearly faster than feature computation by neural networks.

Method	SIFT	ImageNet CNN	Unsup. CNN
Time	$2.95\text{ms} \pm 0.04$	$11.1\text{ms} \pm 0.28$	$37.6\text{ms} \pm 0.6$

Table 1: Feature computation times for a patch of 91 by 91 pixels on a single CPU. On a GPU, the convolutional networks both need around 5.5ms per image.

4 Conclusions

The comparison allows us to draw several conclusions:

1. Both CNNs outperform SIFT on the descriptor matching task. This performance gain is approximately as high as the improvement of SIFT over simple RGB patches.
2. While class labels provided during the neural network training are beneficial when the features are used for classification, the unsupervised CNN training is superior for descriptor matching.
3. The experiment on blurring transformations indicates a limitation of the CNN trained on ImageNet. With the unsupervised network we showed that this can be handled to some extent by incorporating blurred images during training. However, blur is still the most difficult transformation for neural nets, which may indicate some general weakness in the architecture.
4. The computational cost is in favor of SIFT.

While SIFT is still interesting for tasks were speed and simplicity are of major importance, for all computer vision tasks that rely on descriptor matching it is worth considering the use of features trained with convolutional neural networks. The unsupervised training introduced by Dosovitskiy et al. in [5] seems particularly promising.

Acknowledgments

The work was partially funded by the ERC Starting Grant VideoLearn.

References

[1] Bay, H., Ess, A., Tuytelaars, T., Van Gool, L.: Speeded-up robust features (surf). Comput. Vis. Image Underst. 110(3), 346–359 (Jun 2008) 1

[2] Brox, T., Malik, J.: Large displacement optical flow: descriptor matching in variational motion estimation. IEEE Transactions on Pattern Analysis and Machine Intelligence 33(3), 500–513 (2011) 1

[3] Dalal, N., Triggs, B.: Histograms of oriented gradients for human detection. In: CVPR. pp. 886–893 (2005) 1

[4] Donahue, J., Jia, Y., Vinyals, O., Hoffman, J., Zhang, N., Tzeng, E., Darrell, T.: DeCAF: A deep convolutional activation feature for generic visual recognition (2013), pre-print, arXiv:1310.1531v1 [cs.CV] 1, 2

[5] Dosovitskiy, A., Springenberg, J.T., Brox, T.: Unsupervised feature learning by augmenting single images (2014), pre-print, arXiv:1312.5242v3 [cs.CV], ICLR'14 workshop track 2, 3, 9

[6] Girshick, R., Donahue, J., Darrell, T., Malik, J.: Rich feature hierarchies for accurate object detection and semantic segmentation. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2014) 1, 2

[7] Jia, Y.: Caffe: An open source convolutional architecture for fast feature embedding. http://caffe.berkeleyvision.org/ (2013) 2

[8] Krizhevsky, A., Hinton, G.: Learning multiple layers of features from tiny images. Master's thesis, Department of Computer Science, University of Toronto (2009) 2

[9] Krizhevsky, A., Sutskever, I., Hinton, G.E.: ImageNet classification with deep convolutional neural networks. In: NIPS. pp. 1106–1114 (2012) 1, 2

[10] LeCun, Y., Boser, B., Denker, J.S., Henderson, D., Howard, R.E., Hubbard, W., Jackel, L.D.: Backpropagation applied to handwritten zip code recognition. Neural Computation 1(4), 541–551 (Winter 1989) 2

[11] LeCun, Y., Bottou, L., Bengio, Y., Haffner, P.: Gradient-based learning applied to document recognition. Proceedings of the IEEE 86(11), 2278–2324 (November 1998) 2

[12] Lowe, D.G.: Distinctive image features from scale-invariant keypoints. IJCV 60(2), 91–110 (Nov 2004) 1

[13] Matas, J., Chum, O., Urban, M., Pajdla, T.: Robust wide baseline stereo from maximally stable extremal regions. In: Proc. BMVC. pp. 36.1–36.10 (2002), doi:10.5244/C.16.36 4

[14] Mikolajczyk, K., Schmid, C.: A performance evaluation of local descriptors. IEEE Trans. Pattern Anal. Mach. Intell. 27(10), 1615–1630 (2005) 3, 4

[15] Mikolajczyk, K., Tuytelaars, T., Schmid, C., Zisserman, A., Matas, J., Schaffalitzky, F., Kadir, T., Gool, L.J.V.: A comparison of affine region detectors. IJCV 65(1-2), 43–72 (2005) 2, 4

[16] Razavian, A.S., Azizpour, H., Sullivan, J., Carlsson, S.: CNN features off-the-shelf: an astounding baseline for recognition (2014), pre-print, arXiv:1403.6382v3 [cs.CV] 1, 2

[17] Zeiler, M.D., Fergus, R.: Visualizing and understanding convolutional networks (2013), pre-print, arXiv:1311.2901v3 [cs.CV] 1, 2