Convolutional Neural Networks Learn Compact Local Image Descriptors

Christian Osendorfer, Justin Bayer, Sebastian Urban,
and Patrick van der Smagt

Technische Universität München, Fakultät für Informatik, Lehrstuhl für Robotik
und Echtzeitsysteme, Boltzmannstraße 3, 85748 München
osendorf@in.tum.de, bayer.justin@googlemail.com, surban@tum.de, smagt@tum.de

Abstract. We investigate if a deep Convolutional Neural Network can learn representations of local image patches that are usable in the important task of keypoint matching. We examine several possible loss functions for this correspondance task and show emprically that a newly suggested loss formulation allows a Convolutional Neural Network to find compact local image descriptors that perform comparably to state-of-the-art approaches.

Keywords: Convolutional Neural Networks, Non-linear Dimensionality Reduction, Local Image Descriptor Learning.

1 Introduction

Local image descriptors are an important component of many Computer Vision algorithms. They are central to a wide range of Computer Vision tasks like tracking, stereo vision, panoramic stitching, structure from motion or object recognition. Given these widely differing types of use cases, a local image descriptor should be invariant to image, appearance, viewpoint and lightning variations of a local image patch.

Over the last decade many different descriptors have been developed. Several of these are hand designed, with SIFT [1] being the most popular example. In recent years these engineered descriptors where accompanied by approaches that are based on discriminant learning techniques [2–5]. The general motivation behind these methods is that by exploiting statistical properties of image patches through learning, the resulting descriptor is more robust to the previously mentioned variations an image patch can be exposed to.

Tracing back our work to [6], this paper tries to extend the recent success story of Convolutional Neural Networks [7–9] to learning compact local image descriptors. In a series of experiments, we investigate various aspects (different cost functions, different non-linearities, depth) of models based on Convolutional Neural Networks. It turns out that with the correct cost function, Convolutional Neural Networks find compact image descriptors that perform competitively or even better than state-of-the-art algorithms on a challenging benchmark for keypoint matching [3].

M. Lee et al. (Eds.): ICONIP 2013, Part III, LNCS 8228, pp. 624–630, 2013.

Related Work. Similar to our work, [3–5] rely on supervised learning approaches to find compact local image descriptors. These methods suggest different pooling and selection strategies of gradient-based features to learn discriminant descriptors, utilizing boosting [4] or sparse convex optimization [5].

Most similar to our work is [10]. Like us [10] uses a Convolutional Neural Network to learn an encoding for an image patch. However, [10] investigated the applicability of their learnt descriptors only for planar transformations and only compared their performance to SIFT. As it turns out the objective function to train the whole model used in [10] would not be competitive to state-of-the-art approaches on the challenging dataset used in our paper. And finally, [10] relies on gradient based input features on various scales while our algorithm works directly on pixel intensities.

2 General Learning Architecture

A good description of a local image patch is characterized by the fact that *corresponding* image patches are represented by descriptors that are close-by under some metric. *Correspondence* is thereby defined by the various kinds of invariances listed in the first paragraph of section 1. Clearly, the goal of any learning algorithm in this domain is then to find representations *together* with the accompanying metric that performs well on labeled image pairs (corresponding vs. non-corresponding pairs).

DrLim [6] is a framework for energy based models that learn representations using only such correspondence relationships. We utilize DrLim in order to learn low-dimensional mappings for low-level image patches.

The main idea behind DrLim is to map similar (i.e. corresponding) image patches to nearby points on the output manifold and dissimilar image patches to distant points. DrLim is defined over pairs of image patches, x_1 and x_2. The i-th pair (x_1^i, x_2^i) is associated with a label y^i, with $y^i = 1$ if x_1^i and x_2^i are deemed similar and $y^i = 0$ otherwise. We denote by $d(x_1, x_2; \theta)$ the parameterized distance function between the representations of x_1 and x_2 that we want to learn. Based on $d(x_1, x_2; \theta)$ we define DrLim's loss function $\ell(\theta)$:

$$\ell(\theta) = \sum_i y^i \ell_{\text{pll}}(d(x_1^i, x_2^i; \theta)) + (1 - y^i)\ell_{\text{psh}}(d(x_1^i, x_2^i; \theta)) \tag{1}$$

We denote with $\ell_{\text{pll}}(\cdot)$ the partial loss function for similar pairs (it *pulls* similar pairs together) and with $\ell_{\text{psh}}(\cdot)$ the partial loss function for dissimilar pairs (it *pushes* dissimilar pairs apart). Several possible choices for $\ell_{\text{pll}}(\cdot)$ and ℓ_{psh} (denoted by C_i) are investigated in this text:

– C_1 — the original paper for DrLim [6] defined $\ell_{\text{pll}}(\cdot)$ and ℓ_{psh} as follows:

$$\ell_{\text{pll}}(d(x_1, x_2; \theta)) = c_{\text{pll}} d(x_1, x_2; \theta)^2 \tag{2}$$

$$\ell_{\text{psh}}(d(x_1, x_2; \theta)) = c_{\text{psh}}[\max(0, m_{\text{psh}} - d(x_1, x_2; \theta))]^2 \tag{3}$$

m_{psh} is a push *margin*: Dissimilar pairs are not pushed farther apart if they already are at a distance greater than m_{psh}. c_{pll} and c_{psh} are scaling factors, both set to $\frac{1}{2}$ in [6].

- C_2 — [10] uses the definitions from [11]:

$$\ell_{\text{pll}}(d(x_1, x_2; \theta)) = \frac{2}{Q} d(x_1, x_2; \theta)^2 \tag{4}$$

$$\ell_{\text{psh}}(d(x_1, x_2; \theta)) = 2Q \exp(-\frac{2.77}{Q} d(x_1, x_2; \theta)) \tag{5}$$

The constant Q is set to the upper bound of $d(x_1, x_2; \theta)$.

- C_3 — the exponential loss from [4]:

$$\ell_{\text{pll}}(d(x_1, x_2; \theta)) = \exp(y' d(x_1, x_2; \theta)) \tag{6}$$

$$\ell_{\text{psh}}(d(x_1, x_2; \theta)) = \exp(y' d(x_1, x_2; \theta)) \tag{7}$$

where $y' = 2y - 1$ and y indicates whether a given x_1 and x_2 are a corresponding pair or not, i.e. $y' \in \{-1, 1\}$

- C_4 — in this paper we investigate a combination of a hinge-like loss function for ℓ_{pll} with ℓ_{psh} set as in [6]:

$$\ell_{\text{pll}}(d(x_1, x_2; \theta)) = c_{\text{pll}}[\max(0, d(x_1, x_2; \theta) - m_{\text{pll}})] \tag{8}$$

$$\ell_{\text{psh}}(d(x_1, x_2; \theta)) = c_{\text{psh}}[\max(0, m_{\text{psh}} - d(x_1, x_2; \theta))]^2 \tag{9}$$

m_{pll} is a pull *margin*: Similar pairs are pulled together only if they are at a distance above m_{pll}.

For a complete definition of $\ell(\theta)$ we still need $d(x_1, x_2; \theta)$: for C_1, C_3 and C_4 it is defined as the Euclidean distance between the learned representations of x_1 and x_2:

$$d(x_1, x_2; \theta) = \|f(x_1; \theta) - f(x_2; \theta)\|_2 \tag{10}$$

For C_2 it is defined as

$$d(x_1, x_2; \theta) = \|f(x_1; \theta) - f(x_2; \theta)\|_1 \tag{11}$$

In both cases $f(\cdot)$ denotes the mapping from the (high-dimensional) input space to the low-dimensional representation space. In this paper, $f(\cdot)$ is a Convolutional Neural Network.

2.1 Convolutional Neural Networks

A Convolutional Neural Network [7] is a special kind of neural network for working with images. It is composed of multiple layers, where the output of every

layer is a set of two dimensional arrays called feature maps. A feature map is produced by convolving the respective input with a filter, followed by a non-linear function and a pooling layer. Within the DrLim framework the same network is applied to two different inputs in order to compute the loss for this input pair (see equation 10). Therefore, the architecture is sometimes called a siamese network [12, 13]. In this work we investigate two aspects of a configuration of a Convolutional Neural Network:

- non-linearities: we compare the standard $\tanh(\cdot)$ and the currently often used rectifying linear unit [9].
- depth: we compare models with three and four layers.

3 Experiments

We use the dataset from [3] for evaluating various instances of Convolutional Neural Networks. In contrast to previous approaches actual 3D correspondences, obtained via a stereo depth map, are used for generating this dataset. This allows learning descriptors that are optimized for the non-planar transformations and illumination changes that result from viewing a truly 3D scene. The dataset is based on more than 1.5 million image patches (64×64 pixels) of three different scenes: the Statue of Liberty (about 450,000 patches), Notre Dame (about 450,000 patches) and Yosemites Half Dome (about 650,000 patches). We denote these scenes with LY, ND and HD respectively. There are 250000 corresponding image patch pairs and 250000 non-corresponding image patch pairs available for every scene. We train on one scene and evaluate the learned embedding function on the other two scenes. Evaluation is done on the same test sets (50000 matching and non-matching pairs) used also by other approaches.

We achieve the best results on this benchmark with a Convolutional Neural Network paired with the loss function C_4. The network has 4 convolutional layers[1] and uses the tanh non-linearity. Moreover, Table 1 shows that this Convolutional Neural Network (the entry denoted *CNN1*) performs comparably to other state-of-the-art approaches in terms of the 95% error rate which is the percent of incorrect matches when 95% of the true matches are found: After computing the respective distances for all pairs in a test set, a threshold is determined such that 95% of all matching pairs have a distance below this threshold. Non-matching pairs with a distance below this threshold are considered incorrect matches. Figure 1 shows the ROC curves of *CNN1* for the three different training settings.

In order to avoid unnecessary clutter, we describe only qualitatively the results of comparing different settings for loss functions, non-linearities and depth:

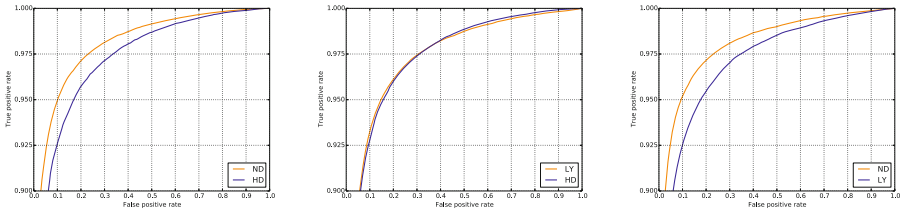
[1] 20 feature maps with kernel size 5×5 followed by a $(2, 2)$ max pooling; a second convolutional layer, again with 20 feature maps and kernel size 5×5 and $(2, 2)$ max pooling; a third convolutional layer, again with 20 feature maps and kernel size 4×4 and $(2, 2)$ max pooling; and a fourth convolutional layer with 64 feature maps and kernel size 5×5.

- Loss functions: C_4 performed at least by $2\% - 3\%$ better than C_1, C_2 or C_3. The idea of having a *pull* margin m_{pll} is crucial for the good performance of C_4. Without it, a noticeable performance drop happens. Interestingly, the results from the original DrLim formulation (C_1) can be improved by utilizing a pull margin, too.
- Non-linearities: Contrary to recent reports [9] on good performance due to linear rectifying units, the networks with a tanh non-linearity performed at least by 5% better than those a the linear rectifying unit.
- Depth: We also tested a Convolutional Neural Network with 3 layers (the total number of parameters was similar to the network with 4 layers). The 4 layer network outperformed this network by approximately $1\% - 1.5\%$.

Table 1. Error rates, i.e. the percent of incorrect matches when 95% of the true matches are found. Every subtable, indicated by an entry in the *Method* column, denotes a descriptor algorithm. The line below every method denotes the size of the descriptor (e.g. 32d denotes a 32 dimensional descriptor). The 128 dimensional SIFT descriptor [1] does not require learning (denoted by $-$ in the column *Training set*). The numbers in the columns labeled LY, ND and HD are the error rates of a method on the respective test set for this scene. [3, 5] do not have results when trained on the LY scene (indicated by $\times$). L-BGM is presented in [4]. *CNN2* is trained on *two* out of the three datasets, see section 3.1

		Test set		
Method	**Training set**	LY	ND	HD
SIFT	–	31.7	22.8	25.6
L-BGM (64d)	LY	–	14.1	19.6
	ND	18.0	–	15.8
	HD	21.0	13.7	–
[3] (29d)	LY	–	×	×
	ND	16.8	–	13.5
	HD	18.2	11.9	–
[5] (29d)	LY	–	×	×
	ND	14.5	–	12.5
	HD	17.4	9.6	–
CNN1 (32d)	LY	–	10.1	17.6
	ND	14.6	–	15.3
	HD	17.6	9.5	–
CNN2 (32d)	LY/ND	–	–	12.3
	LY/HD	–	7.3	–
	ND/HD	13.3	–	–

Every image patch is preprocessed by subtracting its mean and dividing by its standard deviation. All models are trained with standard gradient descent. Training stops when a local minimum of the DrLim objective is reached. We never faced the problem of overfitting (probably because the number of parameters is very small compared to the size of the training set), and thus did not use a validation set. Instead we observed that using a validation set had a negative effect on our final results – the data in the validation set is more useful for actual training. Finally, the hyperparameters for C_4, namely c_{pll}, c_{psh}, m_{pll} and m_{psh} are 0.5, 3, 1.5, and 5 respectively. Notably, these hyperparameters are *not* scene dependent.



(a) Training set: LY (b) Training set: ND (c) Training set: HD

Fig. 1. True Positive Rates and False Positive Rates for *CNN1*. A plot is denoted by its training set and shows the ROC curves on the two remaining test sets. Best viewed in color.

3.1 Data Augmentation

Convolutional Neural Networks benefit from abundant data. A successful method to artificially enlarge the available amount of data is to generate new input data by applying different kinds of transformations to the original dataset [8, 9]. Yet, we did not manage to improve the error rates that we achieve on the original dataset with this approach. However, utilizing data from two scenes improves error rates noticeably: we train on two scenes and evaluate on the remaining one. Following this approach, we are able to improve our error rates by at least 2% (see Table 1, last entry, *CNN2*).

4 Conclusion and Future Work

In this short paper we showed empirically that a standard Convolutional Neural Network, equipped with a suitable loss function, can find compact representations for local image patches: on a challenging dataset for keypoint matching we were able to perform at least as well as state-of-the-art approaches.

The appeal of a simple parametric model like a Convolutional Neural Network is that it does not require any complex parameter tuning or pipeline optimizations

and that it can be integrated into larger systems that can then be trained in an end-to-end fashion [14]. To be more concrete, the 32 dimensional descriptor proposed in this paper can be used to define a dense representation of an arbitrary image. This dense representation is then fed into another Convolutional Neural Network for e.g. image segmentation [15], which can tune the low-level representations for the specific task at hand through straightforward backpropagation.

Acknowledgments. Sebastian Urban was supported by German Research Foundation (DFG) SPP 1527 Autonomes Lernen.

References

1. Lowe, D.: Distinctive image features from scale-invariant keypoints. International Journal of Computer Vision 60(2), 91–110 (2004)
2. Shotton, J., Johnson, M., Cipolla, R.: Semantic texton forests for image categorization and segmentation. In: Proc. CVPR (2008)
3. Brown, M., Hua, G., Winder, S.: Discriminative learning of local image descriptors. Pattern Analysis and Machine Intelligence 33(1), 43–57 (2010)
4. Trzcinski, T., Christoudias, M., Lepetit, V., Fua, P.: Learning image descriptors with the boosting-trick. In: Proc. NIPS (2012)
5. Simonyan, K., Vedaldi, A., Zisserman, A.: Descriptor learning using convex optimisation. In: Fitzgibbon, A., Lazebnik, S., Perona, P., Sato, Y., Schmid, C. (eds.) ECCV 2012, Part I. LNCS, vol. 7572, pp. 243–256. Springer, Heidelberg (2012)
6. Hadsell, R., Chopra, S., LeCun, Y.: Dimensionality reduction by learning an invariant mapping. In: Proc. CVPR (2006)
7. Jarrett, K., Kavukcuoglu, K., Ranzato, M., LeCun, Y.: What is the best multi-stage architecture for object recognition? In: Proc. ICCV (2009)
8. Ciresan, D., Meier, U., Schmidhuber, J.: Multi-column deep neural networks for image classification. In: Proc. CVPR (2012)
9. Krizhevsky, A., Sutskever, I., Hinton, G.: Imagenet classification with deep convolutional neural networks. In: Proc. NIPS (2012)
10. Jahrer, M., Grabner, M., Bischof, H.: Learned local descriptors for recognition and matching. In: Computer Vision Winter Workshop (2008)
11. Chopra, S., Hadsell, R., LeCun, Y.: Learning a similarity metric discriminatively, with application to face verification. In: Proc. CVPR (2005)
12. Becker, S., Hinton, G.E.: Self-organizing neural network that discovers surfaces in random-dot stereograms. Nature 355(6356), 161–163 (1992)
13. Bromley, J., Bentz, J.W., Bottou, L., Guyon, I., LeCun, Y., Moore, C., Säckinger, E., Shah, R.: Signature verification using siamese time delay neural network. International Journal of Pattern Recognition and Artificial Intelligence 7(4), 669–688 (1993)
14. Hadsell, R.: Learning long-range vision for an offroad robot. PhD thesis, New York University (2008)
15. Schulz, H., Behnke, S.: Learning object-class segmentation with convolutional neural networks. In: Proc. ESANN (2012)