Learning Convolutional NonLinear Features for K Nearest Neighbor Image Classification

Weiqiang Ren*, Yinan Yu†, Junge Zhang*, Kaiqi Huang*
*Center for Research on Intelligent Perception and Computing
National Laboratory of Pattern Recognition (NLPR), CASIA
P.O. Box 2728, Beijing, 100190, P.R. China
Email: {wqren, jgzhang, kqhuang}@nlpr.ia.ac.c
†Baidu Inc.
Email: yuyinan@baidu.com

***Abstract*—Learning low-dimensional feature representations is a crucial task in machine learning and computer vision. Recently the impressive breakthrough in general object recognition made by large scale convolutional networks shows that convolutional networks are able to extract discriminative hierarchical features in large scale object classification task. However, for vision tasks other than end-to-end classification, such as K Nearest Neighbor classification, the learned intermediate features are not necessary optimal for the specific problem. In this paper, we aim to exploit the power of deep convolutional networks and optimize the output feature layer with respect to the task of K Nearest Neighbor (kNN) classification. By directly optimizing the kNN classification error on training data, we in fact learn convolutional nonlinear features in a data-driven and task-driven way. Experimental results on standard image classificatioin benchmarks show that the proposed method is able to learn better feature representations than other general end-to-end classification methods on kNN classification task.**

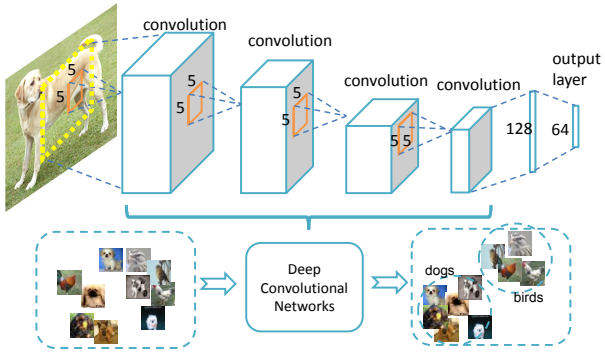


Fig. 1: Learning nonlinear metric for kNN classification with Convolutional Networks.

I. Introduction

Learning low-dimensional feature representations is always a crucial task in machine learning and computer vision. With powerful and discriminative features, we can achieve good performance with very simple classifiers that is comparable to those obtained by more sophisticate and complex models. Within those naive models, K Nearest Neighbor (kNN) algorithm is a simple yet powerful classification algorithm that works surprisingly well in many pattern recognition and computer vision tasks. However, there is an important assumption in kNN, that is, two instances of the same category should be close in Euclidean space. The bad news is that this assumption usually does not hold in real data applications. Take the image classification task for an example, due to the great variations in object size, view, deformation and background clutters, directly compute the Euclidean distance on the raw pixels and perform kNN classification based on this similarity measure will not work well in most real cases. Even with more carefully designed feature representations, such as SIFT [1], HOG [2], and Bag-of-Words [3], the solution is not necessarily optimal as it is not tailored to our training data and our end-goal of kNN classification.

One way to improve the performance of kNN is learning more suitable similarity metrics in a data driven paradigm, which is commonly referred to as metric learning [4], [5] in the machine learning literature. Metric learning aims at learning a similarity/distance metric that pushes data instances of different classes away from each other while pulls data instances of the same class into a compact neighborhood domain. Mahalanobis distance learning is extensively studied in the metric learning field. The Mahalanobis distance between data point $\mathbf{x}$ and $\mathbf{y}$ can be expressed as $d_M(\mathbf{x},\mathbf{y}) = \sqrt{(\mathbf{x}-\mathbf{y})^T\mathbf{M}(\mathbf{x}-\mathbf{y})}$, where $\mathbf{M}$ is a semi-definite positive matrix that to be learned in an optimization algorithm. By decomposition of $\mathbf{M} = \mathbf{A^T A}$, the Mahalanobis distance can be easily interpreted as follows: first project data $\mathbf{x}$ and $\mathbf{y}$ into a new space with linear transformation $\mathbf{A}$; then the Euclidean distance is computed between the projected data. In this way, Mahalanobis distance learning is in fact equivalent to learning a linear transformation. However, due to the complexity of real image data, linear transformation is usually not sufficient to capture the variations in the data. Kernelization is a commonly used method to turn a linear approach to its non-linear counterpart. In general, kernel methods performs better than their linear ones. However, the computational cost is higher and more designation efforts are needed for suitable kernels.

An alternative way is to learn better feature representations that directly tailored to the kNN classification task. From the feature learning point of view, we can adopt more powerful deep models and train the model with respect to our final goal. There have been a great deal of deep models proposed [6], to name a few, RBM [7], Auto-Encoder [8], DBN [7] as well as Convolutioinal Networks [9]. Recently, deep convo-

lutional networks have made great breakthrough in object recognition task, such as object classification [10], [11], object detection [12] and semantic segmentation [13]. With sufficient labeled training data, such supervised deep models successfully learn semantically meaningful feature representations from raw data.

In this paper, we propose to learn low-dimensional feature representations by exploiting the powerful convolutional networks, as is illustrated in Fig. 1. In order to discriminatively train the networks, we adopt the neighborhood component analysis (NCA) [14] algorithm as our supervised loss function. As is pointed out by [14], NCA optimizes the expected leave-one-out classification error of kNN, which forces the networks learn to work well under kNN classification. Another good property of NCA loss is that it is differentiable, making it possible to efficient learn the parameters of the network.

The outline of the remainder of this paper is as follows. In Section II, we discuss the related work on metric learning and representation learning. Section III presents our proposed learning method with deep convolutional networks, which aims to provide an effective feature embedding model that is amenable to kNN classification. In Section IV we present the experimental results on two image classification datasets. Section V draws the conclusion of this paper and discusses the potential applications and future work.

II. Related work

In this section, we discuss the related work as well as their differences to our model.

The topic of learning similarity/distance metrics and feature representations has been widely studied for many years. Mahalanobis metric for clustering (MMC) [15] is the first metric learning algorithm proposed by Xing et al.. By formulating the problem as a convex optimization problem, their algorithm is able to find the global optima using a gradient ascent and iterative projection algorithm. Although obtains good results on clustering problem, MMC is not especially suitable for kNN classification as the model tries to minimized the distance between all similar pairs, without considering neighborhood relationship. Relevant component analysis (RCA) [16] learns the Mahalanobis distance on the so called chunklet, which is a small set of data points with identical but unknown label. RCA compresses the data along the dimensions of the highest irrelevant variability by assigning high weights on relevant dimensions and low weights on irrelevant dimensions. Information-theoretic metric learning (ITML) [17] learns the Mahalanobis distance by minimizing the LogDet divergence subject to linear constraints. [18] proposes a Hamming distance metric learning algorithm that is able to directly learn Hamming binary codes by optimizing a piecewise smooth upper bound on empirical loss.

Neural networks are also used to learn similarity metric and feature embedding. [19] presents a method of convolutional networks for discriminatively learning a similarity metric for face verification. The algorithm is formulated as an energy based model (EBM) and the loss function is defined as the L1 distance between the projected data for a pair of examples. DrLim [20] further extends this model to the dimensionality reduction problem. The loss function of DrLim is defined so that similar examples tend to be closer and dissimilar examples tend to be further from each other. This model is able to optimize the convolutional feature transformer with respect to the metric learning objectives. The learned model projects raw image data onto a low-dimensional feature space which can be further used in other vision tasks such as classification and recognition. However, these algorithms are designed for the face verification and dimensionality reduction problems, thus the learned features are not optimal for kNN problem.

There have been many works that concentrate on learning specific metrics for kNN classification. Neighborhood Components Analysis (NCA) [14] is designed to optimize a stochastic variant of the leave-one-out kNN score on the training set. In NCA, the learning algorithm uses mahalanobis distance, indicating that NCA is in fact a model learning a linear transformation.

Inspired by the idea of NCA, large margin nearest neighbor (LMNN) [21] aims at learning a Mahalanobis distance for kNN classification. LMNN incorporates the large margin criteria of support vector machine into their metric learning framework. The main idea of LMNN is to design a new loss function so that for a pre-defined neighborhood region containing K positive examples, it tries to push the negative samples out of this region. Compared to the non-convex NCA algorithm, LMNN formulates the problem as a convex optimization problem and is solved by semi-definite programming (SDP). Although LMNN achieves good performance for kNN task, it is essentially a linear model learning the Mahalanobis distance.

In order to better model the complex real image data, Salakhutdinov et al. extend NCA to nonlinear NCA [22] by learning a deep neural network. Nonlinear NCA algorithm contains two important stages, namely the pretraining stage and the fine-tuning stage. In the pretraining stage, Restricted Boltzmann Machine (RBM) is trained layer by layer in an unsupervised manner. After that, the stacked RBMs are stacked to form a deep neural network, which is further fine-tuned in a supervised way. The main differences between our model and nonlinear NCA are that our model learns discriminatively trained convolutional nonlinear features and models natural images better. For natural images of normal size, RBM has to work at patch level to reduce the dimensionality of the input space. The convolutional networks can be efficiently applied to full image due the parameters sharing. Another advantage is that it preserves the spatial information of 2D image, giving rise to better feature representations. Moreover, convolutional networks usually need not pretrainning procedure, greatly simplifying the development of models. These differences also indicate that our model is able to handle more complex image data and larger dataset for kNN classificaton.

There are many other metric learning and feature embedding algorithms that are not covered in this section, we refer the reader to [4], [5] for more detailed discussion.

III. Proposed Method

A. Convolutional nonlinear NCA

Convolutional networks [9] is motivated by the Hubel and Wiesel' simplex and complex cell model of cat's visual cortex. In most cases, convolutional networks are discriminatively

trained in a supervised way, using softmax cost or radius basis function classifiers. Afterwards the learned intermediate layer features can be used in other vision applications, such as image retrieval, ranking, object detection as well as semantic segmentation. However, these end-to-end frameworks are generally trained with respect to the classification task, thus suboptimal to these specific problems, e.g., the kNN classification. In order to learn feature embedding that performs well for the kNN classification task, we should build new training objectives that directly optimize kNN classification error.

In this paper, we choose neighborhood component analysis (NCA) [14] as our training loss function. Given N pairs of training examples $\{(\mathbf{x}_i, y_i)|i = 1, \cdots, N\}$, for an example $\mathbf{x}_i$, the probability that example $\mathbf{x}_j$ and $\mathbf{x}_i$ belong to the same category is defined as

$$p_{ij} = \frac{e^{-\|F(\mathbf{x}_i)-F(\mathbf{x}_j)\|^2}}{\sum_{k\neq i} e^{-\|F(\mathbf{x}_i)-F(\mathbf{x}_k)\|^2}} \tag{1}$$

Where $F(\cdot)$ is the nonlinear transformation parameterized by deep convolutional networks. From Eqn. 1, the probability that example $\mathbf{x}_j$ and $\mathbf{x}_i$ fall into the same class is inversely proportional to the Euclidean distance between the learned features.

In kNN classification, the decision of the category for a test example is determined by counting its K nearest neighbors and find the most frequently present class. For NCA, the probability that example $\mathbf{x}_i$ belonging to class c correctly classified can be defined as

$$p_i = \frac{1}{N} \sum_{y_j=c} p_{ij} \tag{2}$$

Then the expected error can be expressed as

$$\begin{aligned} e_{NCA} &= 1 - \frac{1}{N} \sum_{i=1}^{N} p_i \\ &= 1 - \frac{1}{N} \sum_{i=1}^{N} \sum_{y_j=c} p_{ij} \\ &= 1 - \frac{1}{N} \sum_{i,j=1}^{N} p_{ij} y_{ij} \end{aligned} \tag{3}$$

where $y_{ij} = 1$ if $y_i = y_j$, and $y_{ij} = 0$ if $y_i \neq y_j$. The NCA loss function defined in Eqn. 3 is differentiable and continuous, making it suitable for gradient based optimization in neural network training.

Neighborhood component analysis optimizes the expected Leave-One-Out(LOO) classification error which can be seen as a stochastic variant of kNN. By stochastic neighborhood assignment in Eqn. 1, the model is forced to adapt to the local structure of the data distribution and improve the kNN classification performance.

B. Model Learning

In this section we present the details of learning the convolutional network with respect to the NCA loss function so that the learned feature embedding will perform well for kNN classification. Denote the convolutional network as a complex nonlinear transformation $F(\mathbf{x}, \Theta)$, where Θ is the parameter of the network.

Differentiating loss e_{NCA} with respect to network parameters Θ and using the chain rule,

$$\frac{\partial e_{NCA}}{\partial \Theta} = \frac{1}{N} \sum_{i=1}^{N} \frac{\partial e_{NCA}}{\partial F(\mathbf{x}_i, \Theta)} \frac{\partial F(\mathbf{x}_i, \Theta)}{\partial \Theta} \tag{4}$$

The second term in Eqn. 4 can be easily computed using back-propagation algorithm. The first term is the gradient of NCA error with respect to the output features of the network. As NCA cost is differentiable, we can derive this term as follows

$$\frac{\partial e_{NCA}}{\partial F(\mathbf{x}, \Theta)} = y_{ij}(-p_{ij} D_{ij} + p_{ij}^2 E_i) \tag{5}$$

where $D_{ij} \in R^{N\times d}$ is a matrix with the ith row defined as $2(F(\mathbf{x}_i, \Theta) - F(\mathbf{x}_j, \Theta))$ and the jth row defined as $2(F(\mathbf{x}_j, \Theta) - F(\mathbf{x}_i, \Theta))$. All other elements of D_{ij} is zero. E_i is defined as

$$E_i = \sum_{k=1}^{N} \frac{p_{ik}}{p_{ij}} D_{ij} \tag{6}$$

Having computed the gradient of NCA loss function with respect to the parameters of the convolutional network, we can directly apply gradient-based learning rule to update the model parameters. For learning convolutional network, one of the most practical learning algorithm is the stochastic gradient descent (SGD) algorithm, which learns the network by sequentially processing small mini-batch of the data. As SGD usually converges fast with small memory requirements, we adopt SGD in all our implementations and experiments.

One important problem in learning the convolutional network is that we have to shuffle the dataset during model training so that the mini-batch based learning algorithms is not overfitted to the order of data instances. In our experiments, failing to do this sometimes leads to worse models on the same dataset.

The learning rate is an important parameter that has a direct influence on the training of the model. Far sake of faster learning, we usually initialize the learning rate by finding the largest learning rate that the objective function decreases steadily. In most of the learning process, 0.01 or 0.001 are good candidates for initializing the learning rate. After several epochs of learning, the NCA objective function is likely to stop decreasing. In this cases we interrupt the learning and scale the learning rate by 0.1 and resume the training of the network model.

We monitor the learning process by evaluating the network on a hold-out validation set. Early stopping strategy is adopted to avoid overfitting.

In order to speed up the learning of the network, we also adopt the momentum learning. In all of our experiments, we use 0.9 as the momentum.

IV. EXPERIMENTS

In this section, we evaluate our proposed method for kNN classification on two widely used classification dataset, the MNIST [9] handwritten digit dataset and CIFAR-10 [23] dataset.

A. Network Architectures

Designing the convolutional architecture is an important step in developing neural network based algorithms, as there are many hyper-parameters for the network configuration.

The number of layers in the network has a direct influence on the performance of the model. Deeper models tend to be more powerful and are able to model more complex data. However, deeper models are more difficult to train due to the effect of vanishing grading problem. In our experiments we increase the number of layers gradually and stop when the model begins to show evidences of overfitting. The types of network layer are also an important factor in designing deep convolutional networks. Following the successful models in the literature, we first apply several convolutional layers and pooling layers, then several fully connected layers are appended to the network.

The deep convolutional network we used for MNIST dataset is consisted of eight layers: three convolutional layers each followed by a max-pooling layer, the last two layers are fully connect layers. Each convolutional layer have 32 5×5 filters, and the stride is set to $(1, 1)$. For the max-pooling layer, the window size of the pooling region is 3 with a stride of $(2, 2)$. The first fully connected layer has 128 hidden units, and the last fully connected layer has 64 hidden units serving as the final output feature representations of the network. We use tanh function as the activation function for each neuron layer.

For CIFAR-10 datasets, the network architecture is similar to that of MNIST. The first six layers are the same as MNIST, the seventh layer is a fully connected layer with 200 hidden units, and the last layer is a fully connected layer with 100 hidden units. This network is more complicated than that of MNIST in order to handle the more realistic natural images from CIFAR-10 dataset.

B. Evaluation on MNIST and CIFAR-10

1) MNIST: MNIST [9] is a standard benchmark dataset that is widely used in classification, clustering as well and manifold learning algorithm. We report the test error on the test set in Table I. To illustrate the advantage of learning convolutional nonlinear feature using NCA for kNN classification, we have trained another convolutional network sharing the same architecture with the widely used cross entropy loss. After training the network, the output of the last full connection layer is used as the features for kNN classification. The result is present in Table I with the name "CNN features + 1-NN". As expected, our proposed method outperforms the normal convolutional network trained with softmax classifier. The comparison to other method published on MNIST further shows that by directly optimizing convolutional network with respect to the kNN classification task, our proposed method is able to achieve comparable performance using the simple kNN classifier. Noting that our method also performs better than the nonlinear NCA method [22], which needs unsupervised layerwise pretraining and fine tuning. We ascribe this to the convolutional architecture which models 2D image structure better.

TABLE I: Experimental results on MNIST handwritten digit Recognition

Algorithms	Test Error (%)
AE+wd [24]	1.68
DAE-b [24]	1.57
RBM [24]	1.30
CAE [24]	1.14
Deep AE [8]	1.40
DBN [7]	1.20
Euclidean kNN [25]	2.12
LMNN [25]	1.18
NonLinear NCA (7-NN) [22]	1.01
CNN features + 1-NN	1.28
Ours (1-NN)	0.83
Ours (10-NN)	**0.75**

TABLE II: kNN classification results of the proposed method on CIFAR-10 datasets

Algorithms	Test Accuracy (%)
Sparse AE [27]	73.4
Improved LCC [28]	74.5
Kmeans [27] (1600 features)	77.9
Hamming Metric [18]	78.0
Kmeans [27]+1-NN (1600 features)	57.3
CNN features +1-NN (1600 features)	73.6
Ours (1-NN)	75.2
Ours (10-NN)	**78.3**

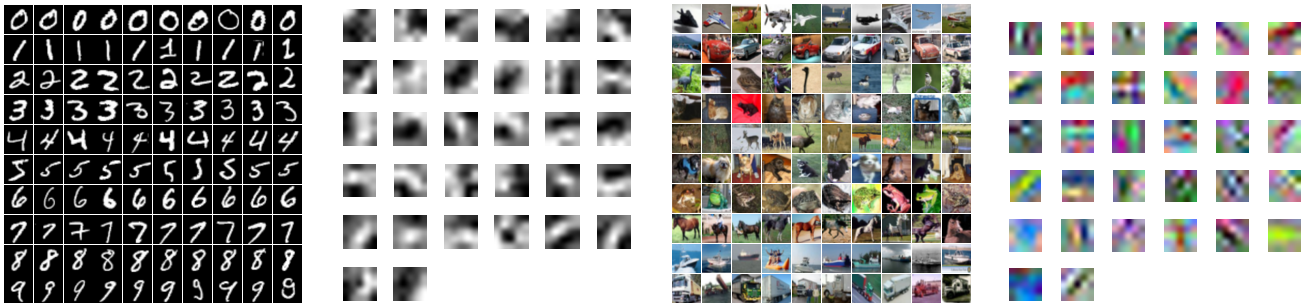
2) CIFAR-10: CIFAR-10 [23] is an established color image dataset collected from the 80 million tiny images [26]. The dataset is composed of 60000 32×32 color images, with 50000 images used for training and 10000 images used for testing. As the CIFAR-10 dataset contains more realistic natural images which is more complex than MNIST, we use a larger convolutional networks. We present the evaluation results in Table II. Noting that since we are evaluating the algorithms under the senario of kNN classification, we do not compare our methods with those using more powerful classifiers for sake of fairness. Again we use convolutional networks trained using softmax as our baseline. It can be seen from Table II that the proposed method achieves better results and is comparable to the state of the art.

C. Analysis and discussion

Quantitative evaluation results on MNIST and CIFAR-10 show that effectiveness of the proposed method. In this section, we analyze the convolutional networks to see what it has learned.

Understanding what the deep convolutional models have learned is a difficult problem and still has a long way to go. Currently one of the most efficient and reliable solution is visualization.

Fig 2 shows the filters learned from MNIST digits and CIFAR-10 dataset. These filters are the weights from the first convolutional layer of our neural networks with the size of 5×5. From the pictures we can conclude that the network



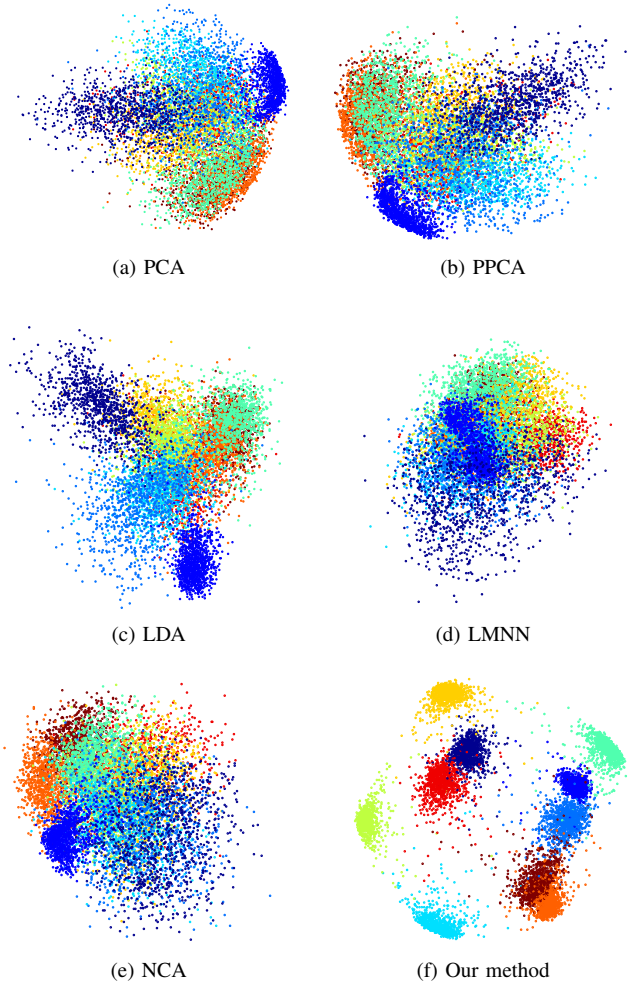
(a) MNIST images (b) MNIST filters (c) CIFAR-10 images (d) CIFAR-10 filters

Fig. 2: 5×5 filters of the first layer learned from MNIST and CIFAR-10 using the proposed algorithm

is able to learn edge/texture like filters that capture low level vision concepts.

In order to illustrate the effect of the proposed learning algorithm more clear, we set the dimensionality of the output layer to 2 and draw the scatter graph of the projected MNIST testing data. For comparison, we also plot the results obtained by principle component analysis (PCA), probability principle component analysis (PPCA), Linear Discriminant Analysis (LDA), Large margin nearest neighbor (LMNN) [21] and Neighborhood Components Analysis (NCA) [14].

From Fig. 3, we can clearly see that global distance learning methods such as PCA, PPCA and LDA perform worse than LMNN, NCA and the proposed method. The reason is that neighborhood relationship is considered so that the learned model is able to adapt to the local data structure. Then it is not surprising that these neighborhood-aware methods works better for kNN classification than other methods. Furthermore, it can be seen that our proposed method performs better than LMNN and NCA, as data of different classes are more clearly separated.



(a) PCA (b) PPCA (c) LDA (d) LMNN (e) NCA (f) Our method

Fig. 3: This figure shows the 2-dimensinal features produced by PCA, PPCA, LDA, LMNN, NCA as well as our proposed method. Data points are MNIST digits and the different colors indicate the classes of the data points.

V. Conclusion

We have present a framework for learning convolutional nonlinear features for K nearest neighbor (kNN) classification. We have shown that by using discriminatively trained deep convolutional networks as representation model, we are able to learn more powerful feature embedding than those linear models and unsupervised models. Furthermore, adopting neighborhood component analysis (NCA) as our loss function has endowed us with the ability to learn feature representations that is specific to kNN classification task. As the kNN classification is widely used in many real applications in computer vision, our model can be used in any neighborhood-aware vision applications to improve the performance of nearest neighbor classifiers. There are still several problems left unsolved. First, the NCA objective is known to be non-convex, making our learning algorithm suffering from local-optima. Another problem is that our learned feature are real valued vectors, which is not very efficient in large scale applications compared with learned binary codes. These problems will be tackled in our future work.

ACKNOWLEDGMENT

This work is funded by the National Basic Research Program of China (Grant No. 2012CB316302), National Natural Science Foundation of China (Grant No. 61322209 and Grant No. 61175007), the National Key Technology R&D Program (Grant No. 2012BAH07B01).

REFERENCES

[1] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *IJCV*, vol. 60, pp. 91–110, November 2004.

[2] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, vol. 1, june 2005, pp. 886 –893 vol. 1.

[3] G. Csurka, C. R. Dance, L. Fan, J. Willamowski, and C. Bray, "Visual categorization with bags of keypoints," in *Workshop on Statistical Learning in Computer Vision, ECCV*, 2004, pp. 1–22.

[4] L. Yang and R. Jin, "Distance metric learning: A comprehensive survey," *Michigan State Universiy*, pp. 1–51, 2006.

[5] A. Bellet, A. Habrard, and M. Sebban, "A survey on metric learning for feature vectors and structured data," *CoRR*, vol. abs/1306.6709, 2013.

[6] Y. Bengio, "Learning deep architectures for ai," *Foundations and Trends® in Machine Learning*, vol. 2, no. 1, pp. 1–127, 2009.

[7] G. Hinton, S. Osindero, and Y. Teh, "A fast learning algorithm for deep belief nets," *Neural computation*, vol. 18, no. 7, pp. 1527–1554, 2006.

[8] Y. Bengio, P. Lamblin, D. Popovici, and H. Larochelle, "Greedy layer-wise training of deep networks," *Advances in neural information processing systems*, vol. 19, p. 153, 2007.

[9] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[10] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems 25*, P. Bartlett, F. Pereira, C. Burges, L. Bottou, and K. Weinberger, Eds., 2012, pp. 1106–1114.

[11] M. D. Zeiler and R. Fergus, "Visualizing and Understanding Convolutional Networks," *ArXiv e-prints*, Nov. 2013.

[12] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," *ArXiv e-prints*, Nov. 2013.

[13] C. Farabet, C. Couprie, L. Najman, and Y. LeCun, "Learning hierarchical features for scene labeling," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, August 2013, in press.

[14] J. Goldberger, S. Roweis, G. Hinton, and R. Salakhutdinov, "Neighbourhood components analysis," in *Advances in Neural Information Processing Systems 17*. MIT Press, 2004, pp. 513–520.

[15] E. P. Xing, A. Y. Ng, M. I. Jordan, and S. Russell, "Distance metric learning, with application to clustering with side-information," *Advances in neural information processing systems*, vol. 15, pp. 505–512, 2002.

[16] N. Shental, T. Hertz, D. Weinshall, and M. Pavel, "Adjustment learning and relevant component analysis," in *Computer VisionECCV 2002*. Springer, 2006, pp. 776–790.

[17] J. V. Davis, B. Kulis, P. Jain, S. Sra, and I. S. Dhillon, "Information-theoretic metric learning," in *Proceedings of the 24th international conference on Machine learning*. ACM, 2007, pp. 209–216.

[18] M. Norouzi, D. Fleet, and R. Salakhutdinov, "Hamming distance metric learning," in *Advances in Neural Information Processing Systems 25*, P. Bartlett, F. Pereira, C. Burges, L. Bottou, and K. Weinberger, Eds., 2012, pp. 1070–1078.

[19] S. Chopra, R. Hadsell, and Y. LeCun, "Learning a similarity metric discriminatively, with application to face verification," in *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, vol. 1. IEEE, 2005, pp. 539–546.

[20] R. Hadsell, S. Chopra, and Y. Lecun, "Dimensionality reduction by learning an invariant mapping," in *In Proc. Computer Vision and Pattern Recognition Conference (CVPR06*. IEEE Press, 2006.

[21] K. Q. Weinberger and L. K. Saul, "Distance metric learning for large margin nearest neighbor classification," *The Journal of Machine Learning Research*, vol. 10, pp. 207–244, 2009.

[22] R. Salakhutdinov and G. Hinton, "Learning a nonlinear embedding by preserving class neighbourhood structure," in *Proceedings of the International Conference on Artificial Intelligence and Statistics*, vol. 11, 2007.

[23] A. Krizhevsky and G. Hinton, "Learning multiple layers of features from tiny images," *Master's thesis, Department of Computer Science, University of Toronto*, 2009.

[24] S. Rifai, P. Vincent, X. Muller, X. Glorot, and Y. Bengio, "Contracting auto-encoders: Explicit invariance during feature extraction," in *Proceedings of the Twenty-eight International Conference on Machine Learning (ICML'11)*, Jun. 2011.

[25] K. Q. Weinberger, J. Blitzer, and L. K. Saul, "Distance metric learning for large margin nearest neighbor classification," in *In NIPS*, 2006.

[26] A. Torralba, R. Fergus, and W. Freeman, "80 million tiny images: A large data set for nonparametric object and scene recognition," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 30, no. 11, pp. 1958–1970, 2008.

[27] A. Coates, H. Lee, and A. Ng, "An analysis of single-layer networks in unsupervised feature learning," *In AISTATS 14*, vol. 1001, p. 48109, 2011.

[28] K. Yu and T. Zhang, "Improved local coordinate coding using local tangents." in *ICML*. Omnipress, 2010, pp. 1215–1222.