Hierarchical Image Representation Using Deep Network

Emrah Ergul[1], Sarp Erturk[1], and Nafiz Arica[2(✉)]

[1] Electronics & Communication Engineering Department of Kocaeli University, Kocaeli, Turkey
{106103002,sertur}@kocaeli.edu.tr
[2] Software Engineering Department of Bahcesehir University, Istanbul, Turkey
nafiz.arica@eng.bahcesehir.edu.tr

Abstract. In this paper, we propose a new method for features learning from unlabeled data. Basically, we simulate k-means algorithm in deep network architecture to achieve hierarchical Bag-of-Words (BoW) representations. We first learn visual words in each layer which are used to produce BoW feature vectors in the current input space. We transform the raw input data into new feature spaces in a convolutional manner such that more abstract visual words are extracted at each layer by implementing Expectation-Maximization (EM) algorithm. The network parameters are optimized as we keep the visual words fixed in the Expectation step while the visual words are updated with the current parameters of the network in the Maximization step. Besides, we embed spatial information into BoW representation by learning different networks and visual words for each quadrant regions. We compare the proposed algorithm with the similar approaches in the literature using a challenging 10-class-dataset, CIFAR-10.

Keywords: Deep network architectures · Image classification · Unsupervised feature extraction · Bag-of-words representation

1 Introduction

The main goal of a learning algorithm is generalization which refers to the ability of having satisfactory performance on the test samples, based on what it has learned in the training phase. At this point of view, our hypothesis function should be robust to bias-variance dilemma [1] which means to construct a learning structure neither too complex for overfitting, nor too simple for underfitting. To do so, we must give our attention mainly to representation learning. This can be described as learning transformations, or posterior distributions in the case of probabilistic models, for the underlying factors of the raw data that extract useful information [2]. Furthermore, we can make the learning algorithm less dependent on the features that are extracted in an unsupervised manner by using huge amount of unlabeled data. Additionally, the source domain may not be the same as or similar to the target domain. This is where Deep Learning Architectures (DLA) proves to be the most successful learning algorithm on the shelf. Unlike hand-engineered feature extraction methods like Scale

V. Murino and E. Puppo (Eds.): ICIAP 2015, Part II, LNCS 9280, pp. 66–77, 2015.
DOI: 10.1007/978-3-319-23234-8_7

Invariant Feature Transform (SIFT) [3], Speeded Up Robust Features (SURF) [4], Pyramid Histogram of Oriented Gradients (PHOG) [5] or GIST [6]; DLA can extract useful features without explicitly using input data statistics.

DLA can be summarized as the multi-layer neural network structures that are formed by the composition of multiple nonlinear transformations of the input data. They aim to have more abstract intermediate features implicitly in deeper layers. By using a deep network, in the case of visual data, one can learn part-to-whole and low level-to-semantic decompositions. What makes DLA effective in representation learning is that they can extract hierarchical features from huge amount of unlabeled data that would prevent overfitting while they are sufficiently complex which might help to recover from underfitting.

In this work, we propose an unsupervised feature learning algorithm for image classification that is based on a deep architecture. Basically, we try to model the feature extraction hierarchically by using unlabeled data through the deep network. The studies based on this basic principle such as [7, 8, 9, 10] use hidden layer activations (i.e. the last layer's activations or the activations of all hidden layers in a concatenated vector) as the new feature vector to be classified. However, we claim that each hidden layer of a deep network corresponds to a different perspective of the same input in another feature space. Therefore, the concatenation alone does not improve the performance adequately. In addition, we do not use the complementary efforts of the previous layers if we consider only the last layer's activations at the supervised classification step. We, on the other hand, implement soft weighted Bag-of-Words (BoW) representations [8], [11], and [12] in the classification by simulating k-means algorithm iteratively for finding visual words at each layer of the deep network. Additionally, we associate spatial layout information by dividing each image into quadrant regions and implementing the model for each region individually. Finally, the image is represented by a pyramid of BoWs.

2 Related Work

The era of the deep networks starts in the 1940's as 'neural networks'. We can describe deep networks as the multi-layer neural structures for data modeling that imitate the most powerful learning bio-machine, brain. One of the key findings has been that the neo-cortex, associated with many cognitive abilities, is layered and hierarchical. It allows sensory signals (i.e. visual, acoustic) to propagate through a complex hierarchy [13] of computational elements (i.e. neurons) that learn to represent observations based on the regularities they expose. The hierarchical nature is that generally the upper layers represent increasingly discriminative representations and are more invariant to transformations such as illumination, scale, rotation and translation. Although it promises to approximate any complex function theoretically, it has not been used widely because of two main restrictions until 2006 when Hinton et al. propose a new approach [10], called 'greedy layer-wise unsupervised pre-training'.

First, traditional gradient based feed forward – back propagation multi-layer networks have a supervised learning objective which refers to the need of labeled data. However, labeled data are often scarce, and the quality of supervision is directly proportional to the experience of human subjects. In short, it is a labor-intensive and application specific job. Given the high expressive power of deep networks, training on insufficient labeled data would also result in overfitting. Another thing is that training a neural network at once involves solving a highly complex and non-convex optimization problem which leads to bad local optima. Because we use nonlinear computational elements (e.g. hyperbolic tangent, sigmoid) in sequential layers and initialize the parameters of the system randomly while trying to minimize them for regularization. As a result, the weights of the earlier layers change slowly, and fail to learn much.

To overcome the problems of supervision and bad local optima, greedy layer-wise unsupervised pre-training is proposed initially in [10] and [14]. The main idea is to learn a hierarchy of intermediate features layer by layer from unlabeled data. We first train a network with only one hidden layer, and only after that is achieved, we start training a network with two hidden layers while keeping the first layer's weights fixed, and so on. Finally, the set of learned layers could be combined to initialize the whole deep network. By using unlabeled data to learn a good initial value for the weights in all the layers, algorithm is now able to learn and discover patterns from massive amount of data while avoiding bad local optima. One can refer to Auto-Encoders (AEs) and Restricted Boltzmann Machines (RBMs) [15] as the primitives or building blocks of the deep learning architectures.

As mentioned above, the deep architectures consist of multiple hidden layers that are assumed to extract more abstract features at their activations when we go deeper. That would be a smart way to 'stack' AEs and RBMs to get discriminative features in the deep architectures. If we stack these two building blocks together, we produce popular deep networks, Stacked Auto Encoders (SAEs) and Deep Belief Networks (DBNs), respectively. To use these structures as classifiers combined with feature learning, we add another layer at the output (i.e. softmax classifier) and may update the parameters of the system at once. These deep networks are implemented in many computer vision problems like object recognition [7], face detection [17] and event detection [18], achieving state-of-art performances.

There is a serious problem with AEs, in that if the hidden layer is similar or greater in size than the input (i.e. over-complete) then the algorithm could simply learn the identity function. It means that we may not get discriminative features. On the other side, we can get higher accuracy in over-complete structures since we make the system more complex. One option is to input corrupted signal and train the system in a way to reconstruct the clean input, which is called De-noising Auto-Encoder (DAE). DAE is successfully implemented in [9] for digit recognition and object classification in a stacked and convolutional manner. Another rational way to get robustness into the system is masking the hidden layer's activations, instead of masking the input for corruption. This is called 'dropout' [19] in the literature where we make a variable amount v of the activations 0. The parameter v represents the percentage of masking to deactivate the outputs of the hidden layer. Finally, we introduce sparsity into the

architecture. In particular, if we impose a sparsity constraint on the hidden units, then the AE will still discover interesting structure in the data, even if the number of hidden units is large [20].

Finally for comparison to our work, Gong et. al. [25] discusses the benefits of deep Convolutional Neural Networks (CNN) especially for retrieval and classification tasks. They argue that Bag-of-Words approach introduces an orderless spectrum which loses the spatial information while CNN depends on too much globally ordered spatial information which lacks especially geometric invariance. To overcome the insufficiencies of the both sides, they combine two approaches at a common platform which is called Multi-scale Orderless Pooling (MOP-CNN). They handle an image at 3 levels for multi-scale representation each of which divides its upper level into quadrants, and the level 1 is the whole image. In this configuration, level 1 is the traditional CNN activations that already preserves the global spatial layout while they implement k-means+VLAD (Vectors of Locally Aggregated Descriptors) [26] algorithm at level 2 and 3 for orderless representations of the patches in order to achieve geometric invariance in a concatenated features vector. For comparison, their start point is a pre-trained seven-layer CNN at each layer, and there is no transitional connection between CNN levels for data representation. Additionally, they use k-means to find centroids for VLAD score aggregation and PCA for dimension reduction. On the other hand, we propose connections between scale levels where the next level uses the hidden layer activations of the previous level for BoW representations; and the last layer represents again the whole image with the composite activations of the quadrants of the previous level.

So far, we have given important details of deep network architectures and their implementations. It is a common practice to use the resulting unsupervised feature representations either as input to a classifier directly, or as initialization for a supervised deep neural network. Alternatively, the outputs of the previous layer may be treated as extra inputs, with the original signal, for the next layer. Although it seems very reasonable to use the hidden layers' activations directly as a new representation, we hypothesize that the system might get poor performance because the activation values are in tight range. Assuming that there is high variance within input data and a limited number of neurons in the hidden layers, the classifier would not approximate the data satisfactorily, leading to underfitting. On the other hand, we may just replicate the data, other than learning useful structures, when we increase the complexity of the system with more neurons, leading to overfitting. To avoid both situations, we need to add randomness or sparsity into the system while adjusting the number of neurons carefully.

Instead of handling aforementioned issues, we introduce a new approach which learns visual words (i.e. code words) hierarchically while training the deep network. We use them to construct BoW representations as a new feature space, not the neuron activations itself. To do so, k-means algorithm is simulated in the deep network. We additionally use spatial information by implementing pyramid-like representations as in [12].

3 Unsupervised Feature Extraction

3.1 Deep Network Architecture

We propose a 3-hidden layer neural network that is trained in a greedy layer wise learning scheme to extract BoW representations at each layer, sequentially. After learning some features at a lower layer, we go further with these learned features and start learning the next layer's weights while keeping the previous ones fixed. Basically, we repeat the same procedure at each layer which is to update the system parameters for finding the fundamentals of a BoW representation, i.e. code words. Fig. 1 summarizes the proposed deep structure visually.

Given a set of unlabeled training images $X=\{x^{(1)}, x^{(2)}, x^{(3)}, ..., x^{(i)}\}$, $x^{(i)} \in R^{nxlxd}$, where n and l are image sizes and d is the color dimension, we first select a large number of patches randomly from the whole image set. We use them as the input signal for the first layer. Each patch has a dimension of w-by-w and has d color channels. So each patch can be represented as a vector of M (i.e. M equals to $w*w*d$) pixel intensity values. But we do not use the intensity values directly as they contain noise and correlation. In our work, we apply contrast normalization that of zero mean-unit standard deviation, and whitening for uncorrelation with the same variance, operations to the intensity values in the preprocessing stage. Given a set of z patch examples, we then define the overall cost function of a typical single-layer network to be:

$$J(W,\ b) = \frac{1}{z}\sum_{j=1}^{z}\frac{1}{2}\left\| h_{W,b}(p^{(j)}) - y^{(j)} \right\|^2 + \frac{\lambda}{2}\sum W^2 \qquad (1)$$

where $h_{W,b}(p)=f(W^T*p + b)$, $f : R \rightarrow R$, *is* the hypothesis function (i.e. the prediction of the system for a sample patch input pattern, p), W and b are the collection of parameters to be minimized, y is the target value and λ is the regularization term. Note that the hypothesis function is nonlinear and non-convex as the activation functions, $f(\cdot)$, are nonlinear; and the parameters are initialized randomly, near to zero. To summarize, we try to find the optimum solution by minimizing the average sum-of-squares error while penalizing the high magnitude of the weights to prevent overfitting. Since this function is derivable, gradient based algorithms can be used to find the optimum parameters and back propagation method is run to calculate derivations.

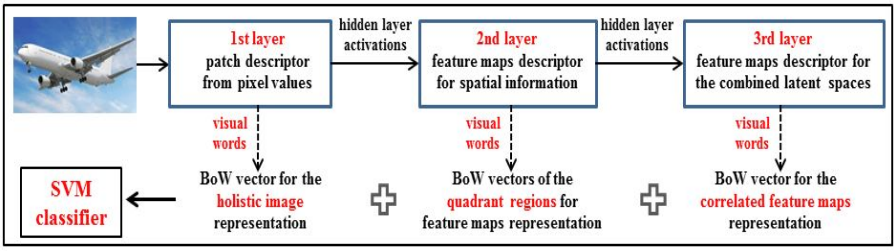

Fig. 1. Overview of the proposed image representation

As we have mentioned before, the AE neural networks optimize the parameters in an unsupervised way by setting the target values to be equal to the inputs, i.e. $y^{(j)}=p^{(j)}$. Our algorithm differs from the auto-encoder method in that we set the target values to the visual words, $c^{(i)}$. So we try to learn code words rather than identity function by simulating k-means algorithm in the network which will be detailed in the next part of this section.

3.2 Simulating K-Means in the Network

We hypothesize that we may achieve representative code words while optimizing the neural network because k-means resembles of the neural network structure in encoding-decoding and reconstruction aspects. So one can transform the input space into another space (i.e. more discriminative) by using the hidden layer activations and continue learning deeper structures while producing BoW representations from visual words for classification tasks. We first take randomly k patch instances as the initial centroids, $c^{(i)}$. Thereafter, we find the nearest centroid for each input and set the target values to the selected centroids, i.e. $y^{(j)}=c^{(i)}$. Now we can run the feed forward – back propagation algorithm in mini batches to update the system parameters by using (1). Note that we keep the centroids, $c^{(i)}$, fixed until we complete all training examples in one epoch. At the end of each epoch, we then update the centroids with the current parameters in the latent space of the current layer:

$$\begin{aligned} &E-step: update \quad (W,b)\ while \quad keeping \quad c^{(i)}\ fixed \\ &\{W_1,b_1,W_2,b_2\} = \arg\min_{W,b} \frac{1}{2z}\sum_{t=1}^{z}\left\|h_{W,b}(p^{(t)})-c^{(i)}\right\|^2 + \frac{\lambda}{2}\sum W^2 \end{aligned} \tag{2}$$

$$\begin{aligned} &M-step: update\ c^{(i)}\ in\ latent\ space, (W_1,b_1) \\ &f_p(p^{(t)};W_1,b_1) = W_1^T p^{(t)} + b_1 \,;\, f_c(c^{(i)};W_1,b_1) = W_1^T c^{(i)} + b_1 \\ &\forall\ p^{(t)} \in P \,;\, q_i^t \leftarrow \left\{ \begin{array}{ll} 1 & if \left\|f_p - f_c\right\| = \min_i \left\|f_p - f_{c^{(i)}}\right\| \\ 0 & otherwise \end{array} \right\} \\ &\forall\ c^{(i)}, i = 1,2,3,\ldots,k \,;\, c^{(i)} = \frac{\sum_t q_i^t\ p^{(t)}}{\sum_t q_i^t} \end{aligned} \tag{3}$$

The activation values of the hidden layer are recorded for both inputs and centroids after updating the network parameters, W and b, in an epoch. Finally, we assign the nearest centroid to each patch in this new domain but we update the centroids in the original input space as in maximization step of (3). These two steps are followed alternately until the cost is under some threshold or the iterations reach to a predetermined number.

3.3 Hierarchical Bag-of-Words Representation

Assuming that we have learned the current layer, we can now produce BoW representations for the input image by using the tuned centroids, $c^{(i)}$. We first extract all patches with a dimension of *w-by-w* in a convolutional way by one-pixel spacing. After preprocessing, we run the network to compute the hidden layer activations which will be used as the input to the next layer. This corresponds to a valid convolution that extracts local patterns in the receptive fields. Additionally, the nonlinear voting scheme of [8] is implemented to the patches of the input image for BoW representations as it is simple to implement, and it offers a softer and sparse encoding.

After extracting global feature maps from patch descriptions for images, we now aim to get local features, again in an unsupervised manner. More formally, we coarsely segment the feature maps, the activations of the first hidden layer, spatially into quadrants and train another specific network for each quadrant. The intuition is that we may extract more abstract feature maps and BoW representations after the first layer, plus spatial information by segmentation is achieved. We follow the same procedure in the second layer. We randomly select patches from the quadrant feature maps and use them for the input to the network, respectively. We finally get another convolutional feature maps that are specific to quadrants while producing new BoW representations. Before going through the third layer, we first join the quadrant feature maps to get an integral input, and then repeat the same procedure. The intuition of the last layer is that we may correlate the spatial information that is extracted from each quadrant individually. Besides, we may get better performance in a holistic representation. Also note that we only whiten the input of the 2nd and 3rd layer as a preprocessing step. The proposed network architecture for hierarchical BoW representations is depicted in Fig. 2.

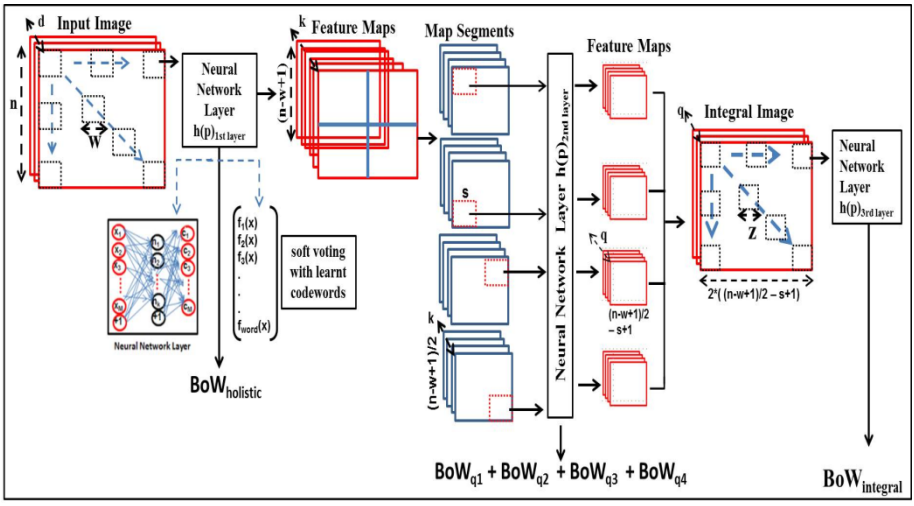

Fig. 2. The proposed three-hidden layer network for hierarchical BoW representations

We can now use the BoW feature vectors layer by layer, in pairs or concatenated at once to represent the instances in a new feature space. This pyramid like structure has been first proposed in [12] and proved to increase the performance. Given these hierarchical BoW representations extracted from the 3-hidden-layer network, we apply standard discriminant algorithms to the labeled training data. In our experiments, we use *L2* Support Vector Machines (SVM) classification method. Cross-validation is implemented to adjust the constant factor (i.e. *C*) of the SVM. The experimental setup and results are explained in the next section.

4 Performance Evaluation

In the experiments, we use a very popular dataset in the literature for object classification, CIFAR-10. The detailed information about this dataset can be found at [22]. It consists of *32-by-32* 50,000 training RBG images which are divided by 5 equal batches, all of them in one of 10 object categories. The test set is already separated and it consists of 10,000 unseen images, and the task is to classify each to its category.

We construct a 3-hidden-layer network for unsupervised feature learning. Code words are found at each layer while optimizing the network parameters. We use the code words for hierarchical BoW representations, and the learned layers to transform the input into feature maps for the next layer. In detail, we use *7-by-7* patches for the first layer, *2-by-2* patches for the second and the third layers as input, with a standard stride of *1* pixel. We feed huge amount of randomly selected patches (i.e. about 500,000) at each layer to optimize the parameters. Notice that we decrease the patch size after the first layer and we have two reasons for doing so. First, the feature maps where each point is *h*-dimensional (i.e. the number of neurons at the previous hidden layer) are fed as the input after the first layer. Increasing the size means increasing the complexity of the network. Second, we need to have sufficiently enough sample patches to create a stable BoW representation, which is to decrease the patch size. But this would also decrease the complexity. So we offset the complexity by adding the mean values of each activation channel within the patch window, like in [17]. Besides, we use hyperbolic tangent function at the hidden layer and no function (i.e. linear) at the output neurons.

Another detail is that we use BoW representations in the classification step while the others [8, 9, 10, 18, 19] use the hidden layer activations directly to produce a new feature vector for the instances. So they usually need an over-complete structure which leads to an extra computational cost. With regards to our approach, the dimensionality of the feature vector depends on the number of code words, not related to the network structure. This is an advantage over the other works. For comparative results, we set the number of centroids sequentially to 100, 200, 400, 800, 1200, 1600 as in [8]; and the test accuracies are displayed in Fig. 3. In the experiments, we use 30 neurons in each layer which are determined by cross-validation and it is much smaller than the input. Finally, it is worth to note that we implement contrast normalization to the first layer's input, and whitening to all layers' inputs as the preprocessing steps.

We first analyze classification results of the BoW representations in pyramid form at Table 1. The experiments are repeated 30 times and the average results are noted. It shows the performance rates at single and multiple layers of the pyramid similar to

format in [12]. The pyramid form refers to the BoW vectors of the previous and the current layers. It is obvious that we get better performances in deeper layers. The best single layer performance is achieved at layer two with 72.16%. We conclude that the spatial information improves the performance when we divide an image into finer sub-regions. Besides, more discriminative representations are produced as we use BoW representations together. The hierarchical BoW representations provide up to 11% increments in classification result when the pyramid forms are used. In overall, 2% increase is achieved when compared to [8] in this particular dataset.

Next, we compare the performance of our method to the similar unsupervised feature learning approaches at Table 2. They mainly use the building blocks that we have already mentioned in section 2; like AEs, RBMs and CNNs. Although our work is much less complex, except k-means [8], we slightly outperform the others at minimum about 1%. This indicates that better performances may be achieved by using the features of all layers together in a computationally efficient way.

Table 1. Classification results of the BoW Representations in pyramid form.

	Pyramid of BoWs W = 1600	
Layer	***Single Layer (%)***	***Pyramid Layers (%)***
1	68.24	---
2	**72.16**	77.45
3	70.17	**79.72**

Table 2. Classification performances of the algorithms on CIFAR-10 test set.

Algorithms	**Accy. (%)**
3-way Factorized RBM [23]	65.3
Convolutional RBM [21]	78.9
Sparse Auto-encoder [8]	73.4
Sparse RBM [8]	72.4
Triangle k-means [8]	77.9
Mean-Covariance RBM [24]	71.0
Convolutional Neural Network [9]	77.5
Concolutional Auto-Encoder [9]	78.2
Proposed 3-layer BoW Encoder	**79.7**

Later, we compare the performances of some methods by changing the dimensionality of features in Fig. 3. It is determined by the number of centroids for the k-means and our approach while it is the number of neurons in the hidden layers for AEs and RBMs. As expected, all algorithms get higher rates by learning more

features. Our work is always better than single layer AE and RBM. On the other side, k-means method outperforms ours by a certain level, 400 features, but we start getting better performances from that point. We can say that the hierarchical representations improve the performance in simple structures.

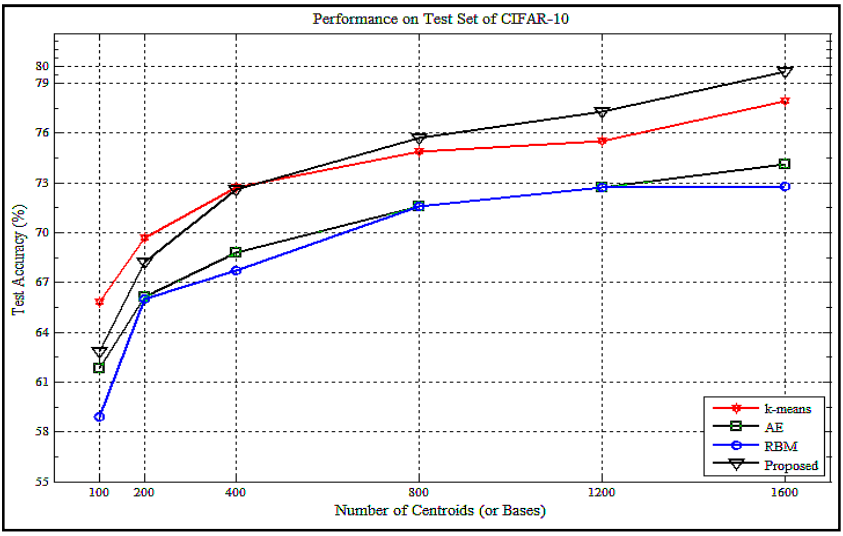

Fig. 3. Comperative results with different number of centroids (or bases)

Finally, we investigate the effect on the overall accuracy of the patches sizes that are used to feed the network. While we keep *2-by-2* size fixed for the second and the third levels, the patch size is changed as the input to the first layer for further evaluations. Also note that stride is still *1* pixel in all levels, the number of centroids is *1600*. At Table 3, we see that the patch size does not significantly impact on the overall performance which is also acknowledged in [8]. Instead, the number of features (i.e. centroids) and the stride size are more effective parameters. The intuition in here is that reasonably high dimensionality for the feature vector mostly introduces better discrimination and we get more samples for the BoW representations if the stride is reduced.

Table 3. The effect of receptive field sizes on the overall accuracy.

Patch Size	Accy. (%)
2	77.4
3	77.6
5	78.9
7	**79.7**
9	79.1
12	78.6

5 Conclusion

In this paper, we analyze the deep network structures in greedy layer-wise unsupervised feature learning. A three-hidden-layer network is learned to produce code words by simulating k-means algorithm which leads to hierarchical BoW representations. Huge amount of unlabeled patch instances are fed to learn single layer parameters and the code words in EM steps. Only after that is done, we start training the next layer while keeping the previous layer's weights fixed, and so on. Note that we do not count on the hidden layer activations directly in classification task.

We gain two basic advantages in this manner. First, the relation between the number of hidden layer neurons and the dimensionality of the feature vectors is broken. Thus one can get more dimensional feature vectors efficiently by using simple networks in our approach. Second, the feature vectors are less constrained since we use code words to achieve BoW vectors. Additionally, we associate location information with the conventional BoW representation. This is accomplished by dividing the input into quadrant regions at the second layer and implementing a network for each subregion individually. In experiments, we see that the performance is increased as we combine the hierarchical BoW representations, leading to outperform more complex approaches in the literature.

For the future work, we plan to embed hierarchical segmentation into this approach for more discriminative code words, and to develop a holistic fine-tuning procedure for updating the parameters after greedy layer-wise training.

Acknowledgements. This work was supported in part by the Scientific and Technological Research Council of Turkey - TUBITAK 2214-B.14.2.TBT.0.06.01-214-83.

References

1. Alpaydın, E.: Introduction to Machine Learning. The MIT Press, London (2004)
2. Bengio, Y., Courville, A., Vincent, P.: Representation Learning: A Review and New Perspectives. PAMI **35**(8), 1798–1828 (2013)
3. Lowe, D.: Distinctive Image Features From Scale Invariant Keypoints. Int'l J. Computer Vision **60**(2), 91–110 (2004)
4. Bay, H., Ess, A., Tuytelaars, T., Gool, L.C.: SURF: Speeded Up Robust Features. Computer Vision and Image Understanding (CVIU) **110**(3), 346–359 (2008)
5. Bosch, A., Zisserman A., Munoz, X.: Representing shape with a spatial pyramid kernel. In: ACM International Conference on Image and Video Retrieval (2007)
6. Oliva, A., Torralba, A.: Modeling the Shape of the Scene: a Holistic Representation of the Spatial Envelope. Int'l J. Computer Vision **42**(3), 145–175 (2001)
7. Krizhevsky, A., Hinton, G.E.: Using very deep auto-encoders for content-based image retrieval. In: ESANN (2011)
8. Coates, A., Lee, H., Andrew, Y.N.: An analysis of single-layer networks in unsupervised feature learning. In: International Conference on Artificial Intelligence and Statistics (AISTATS) (2011)

9. Masci, J., Meier, U., Cireşan, D., Schmidhuber, J.: Stacked convolutional auto-encoders for hierarchical feature extraction. In: Honkela, T. (ed.) ICANN 2011, Part I. LNCS, vol. 6791, pp. 52–59. Springer, Heidelberg (2011)
10. Hinton, G.E., Osindero, S., Teh, Y.W.: A Fast Learning Algorithm for Deep Belief Nets. Neural Computation **18**(7), 1527–1554 (2006)
11. Ergul, E., Arica, N.: Scene classification using spatial pyramid of latent topics. In: ICPR, pp. 3603–3606 (2010)
12. Lazebnik, S., Schmid, C., Ponce, J.: Beyond bags of features: spatial pyramid matching for recognizing natural scene categories. In: Proc. IEEE CVPR, vol. 2, pp. 2169–2178 (2006)
13. Arel I., Rose D.C., Karnowski T.P.: Deep Machine Learning: A New Frontier in Artificial Intelligence Research. IEEE Computational Intelligence Magazine **5** (2010)
14. Bengio, Y., Lamblin, P., Popovici, D., Larochelle, H.: Geedy Layer-wise Training of Deep Networks. NIPS (2007)
15. Bengio, Y.: Learning Deep Architectures for AI. Foundations and Trends in Machine Learning **2**(1), 1–127 (2009)
16. Hinton, G.E.: A Practical Guide to Training Restricted Boltzmann Machine. University of Toronto (2010)
17. Quoc, L., Ranzato, M., Monga, R., Devin, M., Chen, K., Corrado, G., Dean, J., Andrew, N.: Building high-level features using large scale unsupervised learning. In: International Conference in Machine Learning (2012)
18. Yang, Y., Shah, M.: Complex events detection using data-driven concepts. In: ECCV, pp. 722–735 (2012)
19. Srivastava, N.: Improving Neural Networks with Dropout. Master of Science Thesis, University of Toronto (2013)
20. Raina, R., Battle, A., Honglak, L., Packer, B., Andrew Y.N.: Self-taught learning: transfer learning from unlabeled data. In: Proceedings of the 24th Int'l Conf. on Machine Learning (ICML) (2007)
21. Krizhevsky, A.: Convolutional Deep Belief Networks on CIFAR-10. Technical Report (2010)
22. The CIFAR-10 dataset. http://www.cs.toronto.edu/~kriz/cifar.html
23. Ranzato, M., Krizhevsky, A., Hinton, G.E.: Factored 3-way restricted boltzmann machines for modeling natural images. In: ASTATS 13 (2010)
24. Ranzato, M., Hinton, G.E.: Modeling pixel means and covariances using factorized third-order boltzmann machines. In: CVPR (2010)
25. Gong, Y., Wang, L., Guo, R., Lazebnik, S.: Multi-scale orderless pooling of deep convolutional activation features. In: Fleet, D., Pajdla, T., Schiele, B., Tuytelaars, T. (eds.) ECCV 2014, Part VII. LNCS, vol. 8695, pp. 392–407. Springer, Heidelberg (2014)
26. Bergamo, A., Sinha, S.N., Torresani, L.: Leveraging structure from motion to learn discriminative codebooks for scalable landmark classication. In: CVPR (2013)