Improving Visual Feature Representations by Biasing Restricted Boltzmann Machines with Gaussian Filters

Arjun Yogeswaran(✉) and Pierre Payeur

School of Electrical Engineering and Computer Science,
University of Ottawa, Ottawa, Canada
{ayoge099,ppayeur}@uottawa.ca

Abstract. Advances in unsupervised learning have allowed the efficient learning of feature representations from large sets of unlabeled data. This paper evaluates visual features learned through unsupervised learning, specifically comparing biasing methods using Gaussian filters on a single-layer network. Using the restricted Boltzmann machine, features emerging through training on image data are compared by classification performance on standard datasets. When Gaussian filters are convolved with adjacent hidden layer activations from a single example during training, topographies emerge where adjacent features become tuned to slightly varying stimuli. When Gaussian filters are applied to the visible nodes, images become blurrier; training on these images leads to less localized features being learned. The networks are trained and tested on the CIFAR-10, STL-10, COIL-100, and MNIST datasets. It is found that the induction of topography or simple image blurring during training produce better features as evidenced by the consistent and notable increase in classification results.

1 Introduction

Feature representations learned directly from unlabeled data have proven to be more effective than handcrafted features in modern visual object classification applications. Recent advances in unsupervised learning mechanisms have fueled this increased performance. An effective way to assess these methods and their properties is to test them on single-layer networks, as opposed to the commonly-used multi-layered networks used for image classification, such that the influence of network architecture is minimized and the actual performance of the learning mechanism can more easily be isolated and compared. Coates *et al.* [1] structured their comparison similarly by comparing single-layer networks and their classification performance on standard image classification datasets under a variety of parameters. That study outlined the efficacy of several single-layer techniques at learning discriminative features from raw pixel data. Modeled after that study, our current research aims to compare biasing techniques on a single layer network in order to boost image classification performance through generating better features. The properties of these features, which are more

G. Bebis et al. (Eds.): ISVC 2016, Part I, LNCS 10072, pp. 825–835, 2016.
DOI: 10.1007/978-3-319-50835-1_74

difficult to interpret at higher levels in a multi-layer network, may lead to further understanding of their role in object recognition.

In unsupervised learning, Gaussian filters have been considered for inducing topography into feature representations, falling under the idea of incorporating information from neighboring regions [2]. In this work, the same application of Gaussian filters will be performed to create a topography in the network during training. Topography can provide insight into what learned features are similar or are most likely to co-occur. It also allows the grouping of those features through pooling to produce more invariant responses, thus behaving as an improved feature representation. This work will show that, even without pooling, learning features in a topography produces better results on its own than when topography is not used. Topographic independent component analysis (TICA) [3] learns features from unlabeled data and creates topography in those learned features. Kavukcuoglu *et al.* [4] learn invariant features across a topographic map through a technique called predictive sparse decomposition (PSD). These features are learned directly from the data and show better performances when compared to handcrafted features for object classification tasks. The topography also improves the invariance of features compared to regular PSD, with the results showing that they also form a better feature representation. Goh *et al.* [2] introduce 2D topography into a restricted Boltzmann machine (RBM) which learns invariant color features that vary smoothly over the hidden layer.

Downsampling has been used to provide incremental invariance to transformations and dimensionality reduction in the convolutional neural network via pooling [5] or skipping samples [6], thus increasing the discriminative capabilities of the learned features. In a sense, convolution with a low-pass filter produces the same invariance without the dimensionality reduction. That principle is used in this work, where blurring can aid in learning better features, and the Gaussian filter is used here for that purpose.

Gaussian filters provide the link between the two above-mentioned procedures, falling under the idea of combining information from neighboring regions. It is expected that sharing information between neighbors will benefit the learning of good features. Le *et al.* created a deep network that leveraged neighborhoods of features found using TICA en route to state-of-the-art results on a variety of object recognition benchmarks [7]. Sermanet *et al.* learns visual features directly from the data using sparse coding to build a deep network which achieved state-of-the-art results on a variety of pedestrian detection datasets [8]. These results show that better classification results can be achieved when learning features directly from large amounts of image data and provides suitable motivation to explore different methods of biasing unsupervised learning networks to improve feature representations.

This work characterizes the effectiveness of biasing methodologies, specifically based around the Gaussian filter, at learning discriminative features by classification performance on standard datasets. The testing datasets will be the CIFAR-10 [9], STL-10 [1], and COIL-100 [10] image classification datasets. To reduce variability when comparing techniques, an RBM [11] with fixed hyperparameters is used as a singular architecture on which they are tested.

2 Background

2.1 Restricted Boltzmann Machine

The RBM [11] is an undirected bipartite network which uses its hidden layer to represent input data from the visible layer. It is an energy-based model, and calculates the energy of the joint configuration of visible nodes and hidden nodes by (1).

$$E(v,h) = -a'v - b'h - h'Wv. \tag{1}$$

where v and h are the visible and hidden node states, respectively, a and b are the visible and hidden biases, respectively, and W are the symmetric weights connecting the hidden and visible nodes.

Equation (2) determines the probability that a binary hidden node is on, given the visible vector. To deal with image data, the visible vector is modeled using linear nodes. Equation (3) determines the visible node given the hidden vector.

$$P(h_j = 1|v) = sigmoid(b_j + \sum\nolimits_i w_{ij} v_i). \tag{2}$$

$$P(v_i = v|h) = \mathcal{N}(v|a_i + \sum\nolimits_j w_{ij} h_j, 1). \tag{3}$$

where h_j is the jth hidden node, v is the visible node vector, b_j is the bias of the jth hidden node, a_i is the bias of the ith visible node, w_{ij} is the weight connecting the ith visible node, v_i, and h_j, $N(\mu, \sigma^2)$ is a probability density of Gaussian distribution with mean μ and standard deviation σ. Since the image data is normalized, unit variance is used.

Training is accomplished using contrastive divergence (CD), and involves lowering the energy for preferred configurations of hidden and visible nodes, and raising the energy for undesirable configurations [11]. The training alternates between the positive phase and negative phase, where the positive phase samples the hidden state, h^+, and the visible state, v^+, from the data while the negative phase produces the reconstructions of the hidden state, h^-, and the visible state, v^-. The weight update is defined as:

$$\Delta w_{ij} = \gamma[< v_i^+ h_j^+ > - < v_i^- h_j^- >]. \tag{4}$$

where γ is the learning rate, and < . > is the average over a number of samples.

Sparsity has been shown to increase discriminative power and optimize RBM representation of data by forcing only a subset of nodes to represent presented data. Lee *et al.* [12] specify a sparsity target and add a regularization term to encourage activation with the target frequency by increasing or decreasing the bias.

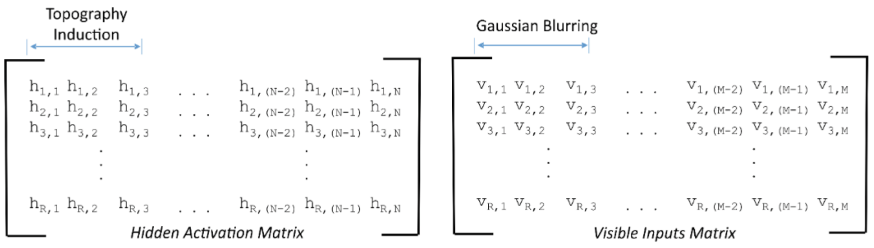
2.2 Gaussian Filters

The purposes of the Gaussian filter in this work are conceptually varied, despite obvious similarities. In the case of the topographic RBM, the Gaussian filter incorporates information from local hidden nodes to induce a dependence thus influencing nodes to develop properties similar to its neighbors. For the purpose of image blurring, the Gaussian filter serves to soften edges such that the network learns blurrier features and perhaps helps remove noise. The effect of the sigma value, which represents the standard deviation of the Gaussian function, will also be evaluated. These filters are normalized to have a gain of 1.

3 Methodologies

The idea of this work is to compare the application of Gaussian filters to induce different effects during unsupervised learning of visual features in an RBM. The compared biasing methods will be: regular RBM, topography, and input blurring.

When training in a batch, the weighted sums in (2) and (3) are performed by matrix multiplication. The visible input matrix contains all of the training data to be used in the current batch, where each row represents a training pattern, or image patch in this work, and each column represents a visible node. The weight matrix contains the weights connecting each visible node to each hidden node. The hidden activation matrix is calculated by multiplying the visible matrix and the weight matrix, resulting in a matrix of activations with each row representing a pattern and each column representing a hidden node. Figure 1 shows how a Gaussian filter is applied in the hidden activation matrix and visible inputs matrix for each method.



$$\begin{bmatrix} h_{1,1} & h_{1,2} & h_{1,3} & \cdots & h_{1,(N-2)} & h_{1,(N-1)} & h_{1,N} \\ h_{2,1} & h_{2,2} & h_{2,3} & \cdots & h_{2,(N-2)} & h_{2,(N-1)} & h_{2,N} \\ h_{3,1} & h_{3,2} & h_{3,3} & \cdots & h_{3,(N-2)} & h_{3,(N-1)} & h_{3,N} \\ & \vdots & & & \vdots & & \\ h_{R,1} & h_{R,2} & h_{R,3} & \cdots & h_{R,(N-2)} & h_{R,(N-1)} & h_{R,N} \end{bmatrix} \quad \begin{bmatrix} v_{1,1} & v_{1,2} & v_{1,3} & \cdots & v_{1,(M-2)} & v_{1,(M-1)} & v_{1,M} \\ v_{2,1} & v_{2,2} & v_{2,3} & \cdots & v_{2,(M-2)} & v_{2,(M-1)} & v_{2,M} \\ v_{3,1} & v_{3,2} & v_{3,3} & \cdots & v_{3,(M-2)} & v_{3,(M-1)} & v_{3,M} \\ & \vdots & & & \vdots & & \\ v_{R,1} & v_{R,2} & v_{R,3} & \cdots & v_{R,(M-2)} & v_{R,(M-1)} & v_{R,M} \end{bmatrix}$$

Fig. 1. Application of convolution, where h are hidden node activations, v are visible inputs, M is the # of visible nodes, N is the # of hidden nodes, and R is # of patterns. Gaussian filters for topography induction are applied to the hidden activation matrix (left), while the input blurring filter is applied to the visible inputs matrix (right)

3.1 Regular RBM

An RBM with regularization only to induce sparsity [12] is used as a control and will be compared as a baseline. This will be referred to as the regular RBM.

3.2 Topography

Topography is induced in the RBM by biasing the network during training. By ordering the nodes for a 1D topography, or arranging the hidden nodes in a grid for a 2D or 3D topography, neighboring nodes can be determined. Applying a Gaussian filter to hidden node activations at each example, each node incorporates information about its neighboring nodes. Applied during learning, adjacent nodes develop slightly different features that gradually vary across the grid.

Assuming batch training, a Gaussian filter is applied among adjacent hidden nodes exposed to the same pattern in the activation matrix. The positive phase activations of the hidden nodes are modified by (5) as found in [2].

$$\hat{h}_j^{(k)} = \sum\nolimits_n^N h_n^{(k)+} \omega(j, n). \tag{5}$$

where $\hat{h}_j^{(k)}$ is the topography-induced positive activation of hidden node j at pattern k, $h_n^{(k)+}$ is the positive phase hidden activation of hidden node n at pattern k, and the neighborhood function, ω, is a set of fixed neighborhood weights which controls the impact of the surroundings on each activation. ω is set to a Gaussian function. 1×3, 3×3, and $3 \times 3 \times 3$ kernels are used for the 1D, 2D, and 3D topographies, respectively.

3.3 Input Blurred

For input blurring, image patches are blurred using a Gaussian filter with kernel width 3 and varying sigmas, using (6), before being passed to the RBM.

$$v_i^{(k)} = \sum\nolimits_{m=1}^M v_m^{(k)} \omega(i, m). \tag{6}$$

where $v_i^{(k)}$ is the visible node i at pattern k being blurred, $v_m^{(k)}$ is a neighboring visible node m at pattern k, and the neighborhood function, ω, is the Gaussian filter.

4 Experimental Work

4.1 Preprocessing

In both training and testing, all patches are contrast normalized and whitened, as these are common techniques to reduce redundant information [1].

4.2 Training

CIFAR-10 is composed of real-world 32×32 colour images containing objects belonging to 10 different classes, with a total of 50,000 training images and 10,000 testing images. STL-10 is similar to CIFAR-10 with 10 classes, but contains 96×96

colour images. It contains 500 training images and 800 test images per class, plus 100,000 unlabeled images for unsupervised learning. COIL-100 contains images of 100 objects on a black background rotated 360° about the vertical axis at 5° increments, resulting in 72 images per object. Samples of CIFAR-10 and COIL-100 are shown in Fig. 2.

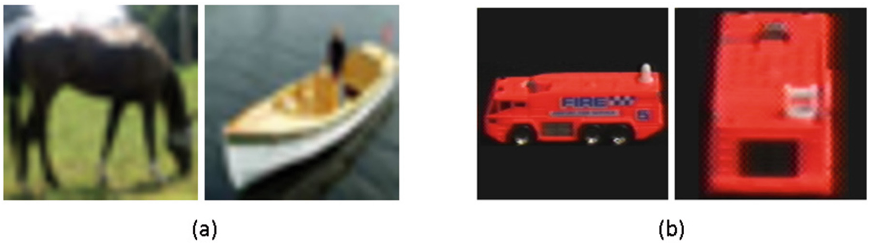


Fig. 2. (a) Some examples of images found in CIFAR-10, with labels of horse and boat. (b) Example object from different angles in the COIL-100 dataset

Primarily, the network is trained on random patches from the CIFAR-10 dataset. 50,000 random 8 $\times$ 8 color patches, divided into batches of 100, were used as training data. The networks were trained for 100 epochs. This dataset contains enough variation to learn a variety of features from real-world images. As an additional experiment, the network is also trained on random patches from STL-10 as well as random patches from COIL-100 to see if the same effects persisted across training datasets. Finally, the networks are also trained on the MNIST handwritten dataset [13], with the full pattern instead of patches, to determine if these techniques work in a different domain.

4.3 Testing

As a measure of how effective the techniques are, the networks are tested on three datasets: CIFAR-10 [9], STL-10 [1], and COIL-100 [10]. The networks are used to extract features via a rudimentary convolution procedure, and a linear classifier is trained on the resulting outputs.

The convolution procedure follows the one outlined in [1]. The training example is transformed into a set of subpatches, each of which are passed through the RBM to generate a set of feature vectors representing the entire image. Experiments in this work use a stride of 1, which means no patches are skipped over and an image of n-by-n pixels, and an input-patch size of w-by-w, produces an (n-w+1)-by-(n-w+1) representation with K features. y_{ij} denotes the K-dimensional representation extracted at position (i, j). A simple pooling mechanism is implemented by dividing the image into 4 quadrants, and the feature vector is reduced from a (n-w+1)-by-(n-w+1)-by-K representation to a 2-by-2-by-K representation by summing the y_{ij}'s in each quadrant. A standard linear classifier is used for classification with the summed feature vectors and the associated label.

Classification accuracy is reported according to the standard testing procedure for each dataset. For CIFAR-10, the features extracted from the training set are used to train the final classifier; the reported classification accuracy is on the test set. STL-10 is divided in to 10 folds, where features extracted from a subset of examples are used for training and testing in each fold. The reported accuracy is the average test accuracy across the folds. With COIL-100, the testing procedure follows that of Mobahi *et al.* [14]. Features extracted from each object at 0, 90, 180, and 270° are used for training. The testing is conducted on all other angles. With MNIST, convolution is not used since the networks are trained on the full image.

5 Results and Analysis

Experiments were carried out with RBMs using the methodologies detailed earlier: Regular, 1D, 2D, and 3D topographies, and input blurred. All networks are regularized with the a sparsity of 0.01, weight decay of 0.002, and no momentum term. CIFAR-10 is used for training unless otherwise stated. An 8×8 color receptive field size was used in all tests, except with MNIST where the full grayscale image is used. Any other parameters specific to each method are outlined in the results.

In the 3D topography, to keep the same number of nodes in each dimension, a different number of total nodes than the 1D and 2D topography was chosen for comparison. The largest cube number that is smaller than the comparison number is chosen as the number of hidden nodes for the 3D topography. This amounts to 216, 343, 512, 729, 1000, and 1331 nodes for the 3D topography corresponding to 225, 400, 625, 900, 1225, and 1600 nodes for the other topographies, respectively.

Figure 3 shows the results of training a RBM after inducing a 2D topography, where each element in the 20×20 grid is a visualization of the feature that each hidden node learns. For example, a vertical feature node responds best to vertical edges, a colored feature node responds best to the displayed color, and a patterned feature node responds best to the displayed pattern. Examples of features learned by the regular and input blurred RBMs, are also shown in Fig. 3.

The regular RBM contains many localized features resembling Gabor filters, acting as very small edge detectors. The 2D topography contains smooth variations among neighboring nodes in both axes, and is made up of similar edge detectors yet they are much larger in size. The features learned by the RBM with input blurring shows more high frequency features with less localization. The topographical RBM also has more high frequency features with less localization, but they look less clean and isolated; while less pleasant to look at, it indicates a more diverse and complex set of correlations.

Primarily, the sigma parameter in the Gaussian filter will determine the impact of the biasing. The classification accuracies on CIFAR-10 of each of the techniques with varying sigma values is shown in Fig. 4 for 225 nodes and 900 nodes. The graphs show that all of the techniques outperform the regular RBM by a significant margin at peak performance. The stronger biasing of the 3D topography provides too much of a constraint among the nodes, causing them to develop poorer features and decrease performance as sigma increases. Otherwise, a larger sigma generally increases

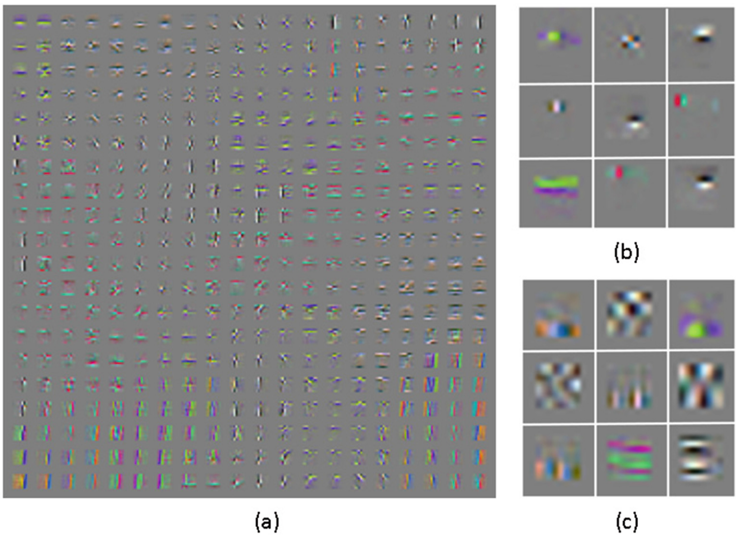


Fig. 3. (a) 20 × 20 grid of features produced by 2D Topography in an RBM. (b) Examples of features learned by the regular RBM. (c) Examples of features learned by the input blurred RBM (Color figure online)

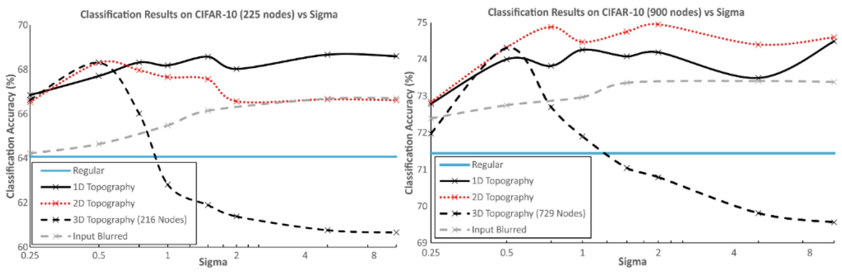


Fig. 4. Classification accuracy on the CIFAR-10 test set vs. Sigma for each technique at 225 hidden nodes (left) and 900 hidden nodes (right)

performance. Aside from the 3D topography, the methods are relatively robust to the sigma parameter choice, and a poor choice is still likely to produce better results than the regular RBM.

The methods and their classification accuracies on CIFAR-10 relative to the number of hidden nodes in the network are shown in Fig. 5. Within each method, the sigma with the best accuracy at 900 nodes is the representative. As expected, an increase in hidden nodes produces an increase in the quality of the feature representation thus increasing the classification accuracy. Here, the input blurring and the topographical methods show a constant improvement over the regular RBM.

The simple blurring of training images produces a surprising increase in classification accuracy, producing over a 2% increase at 225 nodes and 900 nodes. The

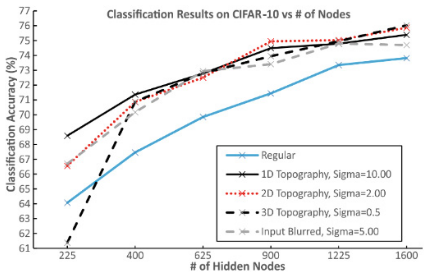


Fig. 5. Best parameters for each method comparing the classification accuracy on the CIFAR-10 test set vs. the number of hidden nodes

biasing to induce topography also produces better classification results than without biasing. At 225 nodes, the 1D topography's performance exceeds the regular RBM's by over 4%. At 900 nodes, the 2D topography's performance exceeds the regular RBM's by over 3%.

Training on CIFAR-10 produces image features which translate well to other visual datasets, including STL-10 and COIL-100. The classification accuracies of each method tested on various datasets, when trained on CIFAR-10 at 1600 nodes, is shown in Table 1. The sigma with the best classification accuracy on CIFAR at 1600 nodes is used, determined by evaluations between 0.0 and 2.0.

Table 1. Classification results, with 1600 nodes, on CIFAR-10, STL-10, and COIL-100, when trained on CIFAR-10. Uses the sigma which produces the best results on CIFAR-10 for each method. Note that the topographic 3D RBM uses 1331 nodes

Method (RBM)	Classification accuracy (%)		
	CIFAR	STL	COIL
Regular	73.82	51.49	84.24
Topography 1D (σ = 1.25)	75.41	52.21	85.69
Topography 2D (σ = 1.00)	**76.24**	54.06	86.28
Topography 3D (σ = 0.75)	75.65	**54.76**	84.28
Input Blurring (σ = 1.50)	74.76	52.10	**86.82**

Again, each of the techniques produces a notable increase in performance over the regular RBM across all datasets; some greater than others depending on the dataset. With such a large amount of features, representational power tends to saturate and classification accuracy plateaus. Therefore, it is interesting that the techniques still produce a fixed increase in classification performance despite hidden layer size.

To evaluate representation learning on different data, Table 2 shows the results of training and testing on the same dataset for STL-10, COIL-100, and MNIST.

It can be seen that the techniques, trained on STL-10, produce similar increases relative to the regular RBM as when they are trained on CIFAR-10. STL-10 contains

Table 2. Classification results, with 1600 nodes, on STL-10, COIL-100, and MNIST when trained on themselves. Uses the sigma which produces the best results on CIFAR-10 for each method. Note that the topographic 3D RBM uses 1331 nodes

Method (RBM)	Classification accuracy (%)		
	STL	COIL	MNIST
Regular	51.55	**78.74**	97.57
Topography 1D (σ = 1.25)	53.22	75.66	**97.97**
Topography 2D (σ = 1.00)	**54.34**	75.59	97.88
Topography 3D (σ = 0.75)	54.31	76.06	97.80
Input Blurring (σ = 1.50)	52.28	78.69	97.38

very similar images to CIFAR-10, including a lot of background imagery. These training samples are taken from real images, as a result, they produce similar performance. But training on a simpler dataset, such as COIL-100, produces worse results. This is likely due to the large amount of uninformative patches, since there is a lot of black space in COIL-100 due to a lack of background. These techniques actually degrade performance from the regular RBM if the dataset contains enough uninformative patches. Given the improved results on MNIST, a dataset where a 0.2% increase in accuracy is considered statistically significant [15], it is visible that the topography techniques also translate to another domain.

In general, the topographic methods perform best, and classification accuracy is boosted even without pooling, as the features themselves benefit from the shared information within the neighborhood during training. At peak sigmas, the 1D topography produces similar results to the 2D and 3D topographies. However, it is computationally simpler, since its 1×3 kernel only requires 2 neighbors. Thus, sharing a small amount of information between nodes produces a large improvement in features, and this improvement does not correlate with the number of neighbors, given that the results of the 3D topography are not much better than the others. For these reasons, the 1D topography is a better solution than the other topographic methods when training time is important. Otherwise, they are comparable in terms of accuracy.

6 Conclusion

This paper performed a comparison between biasing effects of Gaussian filters in the training of RBMs to evaluate their relative performance when it comes to learning good visual feature representations. The quantitative comparison was based on classification results on several image classification datasets.

Primarily, biasing to induce topography quite obviously produces better classification results than without biasing. The 1D topography provides the best balance between computational simplicity and effectiveness. The simple blurring of training images also produces a notable increase in classification accuracy.

The comparison between classification results produces tangible evidence that, despite differences in the approach, improved features can be achieved by biasing the network appropriately. The same biasing can likely be applied to improve image classification results on larger scale multi-layer networks. Overall, this work shows that the simple sharing of information between neighboring nodes, both input and hidden, during training allows the networks to consistently develop better visual feature representations.

References

1. Coates, A., Lee, H., Ng, A.: An analysis of single-layer networks in unsupervised feature learning. In: Proceedings of International Conference on Artificial Intelligence and Statistics, pp. 215–223 (2011)
2. Goh, H., Kusmierz, L., Lim, J.-H., Thome, N., Cord, M.: Learning invariant color features with sparse topographic restricted Boltzmann machines. In: Proceedings of IEEE Conference on Image Processing, pp. 1241–1244 (2011)
3. Hyvärinen, A., Hoyer, P., Inki, M.: Topographic independent component analysis. Neural Comput. **13**, 1527–1558 (2001)
4. Kavukcuoglu, K., Ranzato, M., Fergus, R., LeCun, Y.: Learning invariant features through topographic filter maps. In: Proceedings of IEEE Conference on Computer Vision and Pattern Recognition, pp. 1605–1612 (2009)
5. He, K., Zhang, X., Ren, S., Sun, J.: Spatial pyramid pooling in deep convolutional networks for visual recognition. IEEE Trans. Pattern Anal. Mach. Intell. **37**, 1904–1916 (2015)
6. Simard, P., Steinkraus, D., Platt, J.: Best practices for convolutional neural networks applied to visual document analysis. In: Proceedings of International Conference on Document Analysis and Recognition, pp. 958–962 (2003)
7. Le, Q., Ranzato, M., Monga, R., Devin, M., Chen, K., Corrado, G., Dean, J., Ng, A.: Building high-level features with large scale unsupervised learning. In: Proceedings of International Conference on Machine Learning, pp. 81–88 (2012)
8. Sermanet, P., Kavukcuoglu, K., Chintala, S., LeCun, Y.: Pedestrian detection with unsupervised multi-stage feature learning. In: Proceedings of IEEE Conference on Computer Vision and Pattern Recognition, pp. 3626–3633 (2013)
9. Krizhevsky, A.: Learning multiple layers of features from tiny images. Technical report, University of Toronto (2009)
10. Nayar, S., Nene, S.A., Murase, H.: Real-time 100 object recognition system. IEEE Trans. Pattern Anal. Mach. Intell. **18**, 1186–1198 (1996)
11. Hinton, G., Salakhutdinov, R.: Reducing the dimensionality of data with neural networks. Science **313**, 504–507 (2006)
12. Lee, H., Ekanadham, C., Ng, A.: Sparse deep belief net model for visual area V2. In: Proceedings of Advances in Neural Information Processing Systems, pp. 873–880 (2008)
13. LeCun, Y., Bottou, L., Bengio, Y., Haffner, P.: Gradient-based learning applied to document recognition. Proc. IEEE **86**, 2278–2324 (1998)
14. Mobahi, H., Collobert, R., Weston, J.: Deep learning from temporal coherence in video. In: Proceedings of International Conference on Machine Learning, pp. 737–744 (2009)
15. Larochelle, H., Bengio, S.: Classification using discriminative restricted Boltzmann machines. In: Proceedings of International Conference on Machine Learning, pp. 536–543 (2008)