An analysis of the connections between layers of deep neural networks

Anonymous Author(s)
Affiliation
Address
email

Abstract

We present an analysis of different techniques for selecting connection matrices between layers of deep neural networks. Traditional deep neural networks use random connection tables between layers to keep the number of connections small and tune to different image features. This kind of connection performs adequately in supervised deep networks because their values are refined during the training. On the other hand, in unsupervised learning, one cannot rely on back-propagation techniques to learn the connections between layers. This is because unsupervised networks do not use labeled data, and thus an error signal cannot be computed. In this work, we tested 4 different techniques for connecting the first layer of the network to the second layer on the CIFAR and SVHN datasets and showed that the accuracy may change up to 3% depending on the technique used. We also showed that learning the connections based on the co-occurrences of the features does not confer an advantage over a random connection table in small networks. This work is helpful to improve the efficiency of connections between the layers of unsupervised deep neural networks.

1 Introduction

Most scientists and engineers are fascinated by the design of an artificial vision system that can reproduce some of the human visual system's capabilities of detecting, categorizing, tracking objects in view. The availability of a real-time synthetic vision system with such capabilities would find use in a large variety of applications, such as: autonomous cars, co-robots helpers, smart appliances, cellular-phones, to name a few. The most promising approach in recent years is the fusion of bio-inspired and neuromorphic vision models with machine learning [1–9]. This field, named Deep Learning, has provided state-of-the-art results in the categorization of multiple objects in static frames [10].

Deep Learning networks are computer-vision and computational-neuroscience models of the mammalian visual system implemented in deep neural networks, where each network layer is composed of: linear two-dimensional filtering, non-linearity, pooling of data, output data normalization [7–9]. Deep networks are trained with the abundant image and video content available on the internet and with large labeled datasets. In particular, deep networks need to learn good feature representations for complex visual tasks such as object categorization and tracking of objects in space and time, identifying object presence and absence. These representations usually involve learning the linear filter weight values from labeled and unlabeled input data. Since labeled data is costly and imperfect due to human error [11–13], the recent focus is on learning these features purely from unlabeled input data [14–18]. These recent methods typically learn multiple layers of deep networks by training several layers of features, one layer at a time, with varying complexity of learning models. Traditional work in unsupervised deep learning has focused on learning layer features, and rarely on the

connections between layers. In this paper, we present an analysis of different connections between layers of deep neural networks for general-purpose vision systems.

Recent techniques based on unsupervised Clustering Learning (CL) are especially promising because they use simple learning methods that quickly converge [18, 19]. These algorithms are easy to setup and train and are especially suited for applied research because environment-specific data can be collected quickly with a few minutes of video and the network can be adapted to specific tasks in minutes.

The paper is organized in the following way: Use of clustering learning for learning filters with different connections of deep networks will be explained in the Section 2. The details of the networks including the pre-processing and similarity calculation for learning the receptive field will be described in Section 3. Finally, we will discuss and analyze our results from CIFAR-10 and SVHN datasets in the Section 4.

2 Main idea and contribution

The connections between layers in a deep neural network are very important parameters. It has been shown that in many cases random filters perform only slightly worse than fully trained network layers [20] suggesting that filters, used in the convolutional module of each layer, might not play a dominant role in determining the network performance. Rather, the connections between layers and the combination of features are mainly responsible for the recent success of supervised deep convolutional networks [10]. In these networks the connection between layers is learned from the data using global gradient-descent techniques at large scale. Because training will intimately select the strength of connections, developers of deep networks have traditionally used fully connected layers, custom table connections [1], or random connections [8]. The reason for using custom connection matrices is to reduce the number of computations, and avoid over-fitting problems.

In unsupervised deep networks, one cannot rely on back-propagation techniques to learn the connections between layers. This is because unsupervised networks do not used labeled data, and thus an error signal cannot be computed. Fully connected layers can be used [18, 19] at the expense of more computational time and also reduced performance. The lower performance is attributable to the averaging effect that a fully connected layer has in an unsupervised network, where the connection weights are not adapted to the training data. In order to use unlabeled data to train large deep networks, we investigate different techniques for grouping features from one layer into the features of the next layers, and at the same time learn the filters needed by the next layer. Figure 1 qualitatively explains the main contribution of the paper.

In Figure 1 we show two layers of a deep network: layer N and layer $N+1$. The maps on the left of the figure are the $N1$ output feature maps of the N-the layer in response to an input image. The maps on the right of the figure are the $N2$ outputs of the first convolutional module of layer $N+1$.

We investigate different techniques for grouping feature maps from layer N into "receptive fields (RF)". These receptive fields are the input of one or multiple maps in layer $N+1$. In order to learn the filters for layer $N+1$ convolutional module, we perform CL only with the RF (and not all $N1$ maps). In Fig. 1, we learn $N2$ filters per group for a total of $N2 \cdot G$ maps in layer $N+1$. We form these RF by 4 different methods.

1. (Fanin=1) Grouping each maps individually. In other words, we create $N1$ groups which have only 1 feature map. $G = N1$ and $K = 1$ is used in this case (Fig. 1). In our tests we fixed the number of filters in the first layer to 32, $N1 = 32$. From each group, 16 filters are learned by CL, ($N2 = 16$). Therefore, in total we produce $N2 \cdot G = 32 \cdot 16 = 512$ filters. The size of filters is 1x5x5.
2. (Fanin=2, learned RF) Grouping a small number of layer N features that co-occur in one or multiple sets of feature maps from layer N. In other words, we look for features that are highly activated in the same pixel, and we group these features into the RF. In Fig. 1, these are G groups of K maps. The idea behind this step is to group features that occur often together in the data [21]. The calculation of the similarity of the feature maps is described in Section 3.3. For our tests we group 2 features together from the first layer. We again created 32 groups ($G = 32$ and $K = 2$), and from each group we learned 16

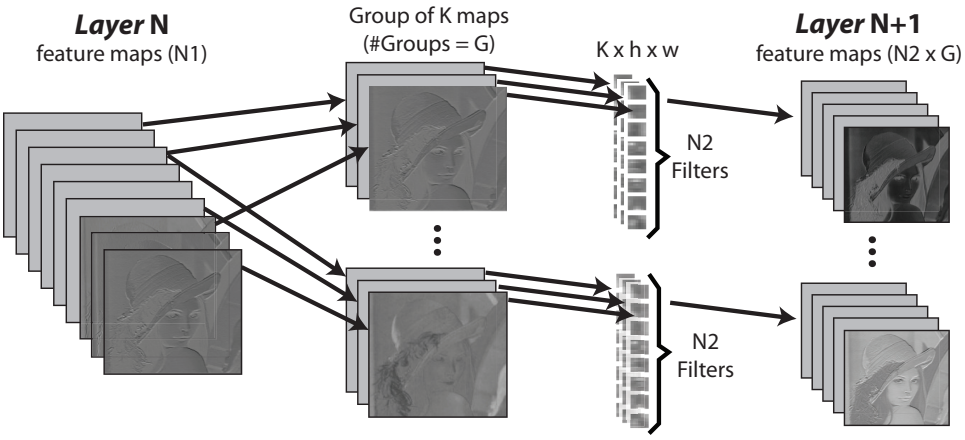


Figure 1: Architecture of the connecting scheme from the first layer to the second layer. Different feature maps from the first layer are grouped together based on metrics (random, similarity, single, and full). Filters are learned from these groups individually by CL. The filters are only applied to the group of feature maps that they are learned from.

filters. Therefore, in total we again produce 512 filters but in this case the filters have size of 2x5x5.

3. (Fanin=2, random RF) Selecting a small number of layer N features randomly. This test is very similar to the second test, but we didn't use any similarity metric to group similar features together. Instead, we group features randomly. All the numbers are kept the same as in the second test. With this test, we want to see if grouping the similar features gives any advantages over the random grouping.
4. (Fanin=32) Grouping all maps into 1 group. This method is basically a full connection from the first layer to the second layer. $G = 1$ and $K = 32$ is used in this case. To keep the number of filters same as the other tests we learned 512 filters from this 1 group which includes all the feature maps from the first layer. In this case, the size of filters is 32x5x5.

In the literature, we found two deep learning papers that are related. Paper [22] groups features on each layer by similarity, but does not explicitly use connection matrices between layers, rather they flatten the data at each layer and re-apply clustering algorithms. This approach is equivalent to forming a full-connection between layers, which is less efficient because each feature is averaged with a large number of other features, thus reducing the signal-to-noise ratio of important features. The main difference between paper [22] and our work is that we focus on learning smaller networks, with much fewer number of filters (32 vs. 4096) in the first and successive layers. The paper [23] presents the interesting idea of learning pooling strategies that include feature maps with similarity in local feature space. This paper uses simple clustering procedures to cluster data in order to improve pooling in the feature space. In this respect, this paper presents similar ideas to the first technique we test in this paper, but is not applied to deep neural networks.

3 Methods

We used the Torch7 software for all our experiments [24], since this software can reduce training and learning of deep networks by 5-10 times compared to similar Matlab and Python tools. In addition it features a "Spatial" mode that allows a trained network to be efficiently applied to any size image, for quick demonstrations and applications.

3.1 Input data

We tested and obtained results using the CIFAR10 [25] and the Street View House Numbers (SVHN) [26] datasets. The SVHN dataset has a training size of 73,257 32x32 images and test size of 26,032 32x32 images. The CIFAR10 dataset has a training size of 20,000 32x32 images and a test size of

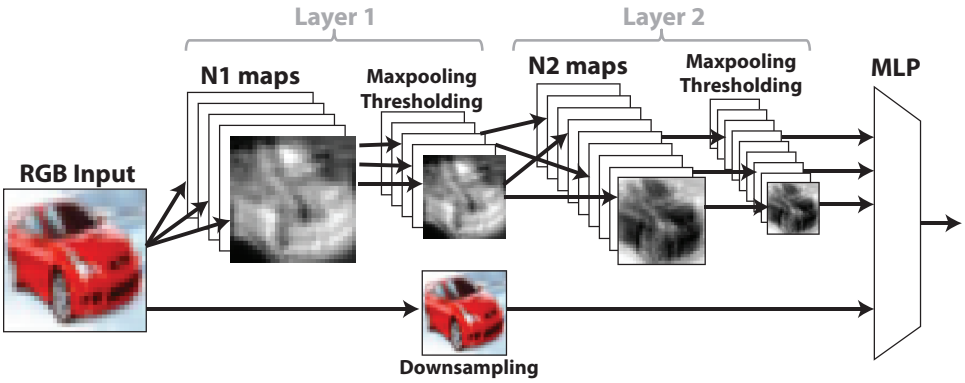


Figure 2: Architecture of the CL network used in this paper: a 2 layer convolutional network with color bypass.

2,000 32x32 images. Both datasets offer a 10 categories classification task on 32x32 size images. The train dataset was in all cases learned to 100% accuracy, and training was then stopped.

We did not use the YUV color space in CIFAR10 and SVHN because they were reporting $\sim 2-4\%$ loss in accuracy. This is contrary to what has been reported by others [7]. We kept the images of CIFAR10 and SVHN in their original RGB format. Input data was first normalized by subtracting out the mean and dividing by the standard deviation and then whitened.

In our experiment we fed all RGB planes to the deep network. We also subsampled the RGB data as much as the deep network. We then concatenated the subsampled RGB data with the output of the deep network into a final vector. This final vector was then passed to a 2-layer MLP classifier.

3.2 Network architecture

We experimented by training an unsupervised deep neural network with 2 layers, not counting pooling and normalization operations. The two layers were composed of a two-dimensional convolutional linear filtering stage, a spatial max pooling stage, and a thresholding stage. The filters of the first two layers are generated with unsupervised clustering algorithms, as explained below. Using the naming convention in [8], for each layer l, x_i is an input feature map, y_i is an output feature map. The input of each layer is a 3D array with n_l 2D feature maps of size $n_{l1} \cdot n_{l2}$. Each component (pixel, neuron) is denoted x_{ijk}. The output is also a 3D array, y_i composed of m_l feature maps of size $m_{l1} \cdot m_{l2}$.

The layers in the clustering learning network used the following sequence of operations:

1. Spatial Convolution module: performing convolutions on images with the learned CL filters: $yc_i = \sum_i k_{ij} * x_i$, where $*$ is the 2D discrete convolution operator.
2. Spatial Max Pooling module: $yp_i = max_{n \times n}(yc_{ij})$ with $n = 2$ in this work.
3. Threshold nonlinearity: $ynl_i = max(ypi_i, 0)$

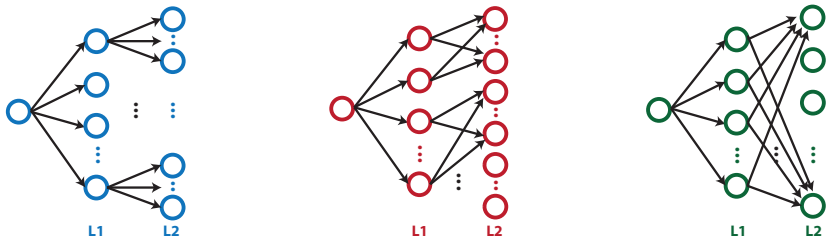
All networks used 32 filters on the first layer, and 512 filters on the second layer. We use k-means clustering algorithm (we refer to this step as *Clustering Learning algorithm*) to learn a set of 32 filters in the first layer,and 512 filters in the second layer. The filter sizes on both layers were set to 5x5 pixels. After the first convolution, 32 filters produced 32 feature maps each size 28x28. Dimension of the feature maps decreased by 2x2 pooling with a stride of 2 pixels. We used max pooling layer which is a bio-inspired approach to reducing the dimensionality of the data at each layer [27]. The output became 32x14x14 pixels. In the second layer, 512 filters produced output of 512x10x10 feature maps which then max pooled again and the size became 512x5x5. We also subsampled the RGB data by 4x4 with a stride of 4 pixels which gave 3x8x8 pixels. We then concatenated the subsampled RGB data with the output of the deep network into a final vector. This final vector was then passed to a 2-layer MLP classifier. The final classifier was fixed to 128 hidden units and 10 output classes for both CIFAR-10 and SVHN.

Table 1: Results on CIFAR-10

Architecture	Accuracy
1 layer (32 filters)	68.8%
2 layer (Fanin=1)	72.8%
2 layer (Fanin=2, random RF)	73.2%
2 layer (Fanin=2, learned RF)	71.2%
2 layer (Fanin=32)	70.0%

Table 2: Results on SVHN

Architecture	Accuracy
1 layer (32 filters)	85.7%
2 layer (Fanin=1)	87.6%
2 layer (Fanin=2, random RF)	88.1%
2 layer (Fanin=2, learned RF)	86.7%
2 layer (Fanin=32)	86.4%



(a) Fan-in = 1 (b) Fan-in = 2 (c) Fan-in = 32

Figure 3: Simplified diagram of networks with different connection techniques.

3.3 Similarity of features

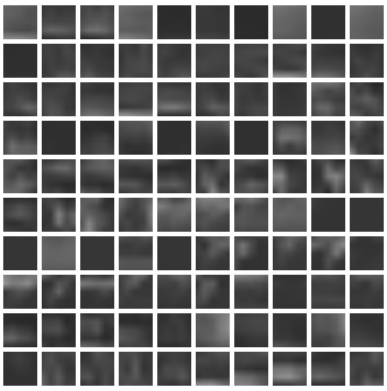
Similarities of features are calculated to learn the RF. The main driving force for this idea is a bio-inspired model of learning in the mammalian brain. It is widely believed that one of the possible physical mechanism of learning in the brain is the Hebbian method [21]. This learning method suggests that a group of pre-synaptic neurons is strongly connected to a post-synaptic neuron if the group (or receptive field, RF) can predict the response of the post-synaptic neuron. In other words, neurons in the RF have strong connections to a neuron in layer $N+1$ if it can predict their response. The learned RF group features that are highly activated together in the same location. This group of highly activated features will then propagate a high value to the same location in the relative map of layer $N+1$. This implements the Hebbian rule. Similarities of features are calculated based on the correlation of the feature maps from the first layer.

4 Results

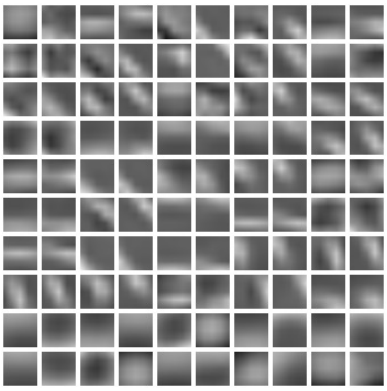
We presented results on CL for general-purpose vision system. We focused on the small number of filter to make our network scalable on hardware. We show results on CIFAR-10 and SVHN datasets with different connection techniques between the first and second layer.

As stated in the Table 1 and 2, 1 layer (32 filters) gives 68.8% on CIFAR-10 dataset and 85.7% on SVHN dataset. The accuracy on the CIFAR-10 is similar to the k-means algorithm with 200 filters from the paper [18]. We are obtaining the similar performance with only 32 filters because we use a different architecture in our network than [18]. We use 2x2 max pooling layer, whereas [18] uses 7x7 subsampling pooling, which gives better results in a small network. Another improvement comes from the downsampling of the image and feeding it to the classifier which increases the accuracy by $\sim 4\%$.

In the table, we first tested single RF architecture. RF of each neuron in the second layer is one feature map from the first layer. Since this architecture is hierarchical and it uses a limited number of connection, it can be interpreted as decision tree as shown in Fig. 3(a). High accuracy of 72.8% compared to 1 layer is achieved here due to the property of decision tree in point that it finds features of features using each filter as a feature descriptor. This method handles only one feature from the previous layer instead of exploiting information from the group of features.



(a) Kernels with full connection.



(b) Kernels with fanin of 2 random connection.

Figure 4: Second layer kernels from CL for comparison.

Next, RF with 2 feature maps are tested to analyze the possible advantages of exploring different features in the same RF. Random combinations and the learned RF's are tested. Learned RF's are obtained based on the similar ideas for learning the receptive fields of complex cells that are reported in several papers in the neural physiology and computational neuroscience literature [21, 28–30]. None of these paper reports results close to the state-of-the-art in publicly available datasets of natural images. We also didn't see any affect of the learning compare to randomization to group the features for each RF.

On the other hand, the second layer with full connection gave significant worse performance than the others. In this case, filters across 32 feature maps are obtained by CL and this architecture suffers from the curse of dimensionality because of the 32x3x5x5 high-dimensional feature space. Fig. 4(a) supports this argument because it does not have distinct edges, rather plain. It was also shown in the paper (cite Adams). Definitely the filters have difficulties in characterizing features in the given dataset.

5 Conclusion

We have presented an analysis of 4 different techniques for selecting connection matrixes between layers of deep neural networks. These techniques consisted of one-to-one connection, two-to-one connections and full-connected networks. For two-to-one connections, we used random grouping as well as grouping based on the similarity of the feature maps from the first layer. We tested these networks on CIFAR and SVHN datasets and showed the fully-connected network performs poorly in unsupervised learning. We also showed that learning the connections based on the co-occurrences of the features does not have an affect over a random connection in small networks. We expect this work to be helpful to improve the efficiency of connections between the layers of deep small neural networks for the further study.

Acknowledgments

We are especially grateful to the the Torch7 developing team, in particular Ronan Collobert, who first developed this great, easy and efficient tool, Clement Farabet, Koray Kavukcuoglu, Leon Bottou. We could not have done any of this work without standing on these giants shoulders. We also thank Soumith Chintala for his help with Torch and the nice discussions.

References

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

[2] R. Hadsell, S. Chopra, and Y. LeCun. Dimensionality reduction by learning an invariant mapping. In *Proc. Computer Vision and Pattern Recognition Conference (CVPR'06)*. IEEE Press, 2006.

[3] K. Gregor, A. Szlam, and Y. LeCun. Structured sparse coding via lateral inhibition. In *Advances in Neural Information Processing Systems (NIPS 2011)*, volume 24, 2011.

[4] M. Riesenhuber and T. Poggio. Hierarchical models of object recognition in cortex. *Nature neuroscience*, 2:10191025, 1999.

[5] T. Serre, A. Oliva, and T. Poggio. A feedforward architecture accounts for rapid categorization. *Proceedings of the National Academy of Sciences*, 104(15):64246429, 2007.

[6] T. Serre and T. Poggio. A neuromorphic approach to computer vision. *Communications of the ACM*, 53(10):5461, 2010.

[7] K. Jarrett, K. Kavukcuoglu, M. Ranzato, and Y. LeCun. What is the best multi-stage architecture for object recognition? In *Proc. International Conference on Computer Vision (ICCV'09)*. IEEE, 2009.

[8] Y. LeCun, K. Kavukvuoglu, and C. Farabet. Convolutional networks and applications in vision. In *Proc. International Symposium on Circuits and Systems (ISCAS'10)*. IEEE, 2010.

[9] Y. Boureau, J. Ponce, and Y. LeCun. A theoretical analysis of feature pooling in vision algorithms. In *Proc. International Conference on Machine learning (ICML'10)*, 2010.

[10] A. Krizhevsky, I. Sutskever, and G. Hinton. ImageNet classification with deep convolutional neural networks. *Advances in Neural Information Processing Systems*, 25, 2012.

[11] A. Karpathy. Lessons learned from manually classifying CIFAR-10. `http://karpathy.ca/myblog/2011/04/27/lessons-learned-from-manually-classifying-cifar-10-with-code/`, April 2011.

[12] A. Torralba and A. A. Efros. Unbiased look at dataset bias. In *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*, page 15211528, 2011.

[13] X. Hou, A. Yuille, and H. Koch. A meta-theory of boundary detection benchmarks. In *NIPS Workshop on Human Computation for Science and Computational Sustainability*, 2012.

[14] B. A. Olshausen and D. J. Field. Emergence of simple-cell receptive field properties by learning a sparse code for natural images. *Nature*, 381(6583):607609, 1996.

[15] A. Hyvärinen and E. Oja. Independent component analysis: algorithms and applications. *Neural networks*, 13(4):411430, 2000.

[16] G. E. Hinton, S. Osindero, and Y. W. Teh. A fast learning algorithm for deep belief nets. *Neural computation*, 18(7):15271554, 2006.

[17] P. Vincent, H. Larochelle, Y. Bengio, and P. A. Manzagol. Extracting and composing robust features with denoising autoencoders. In *Proceedings of the 25th international conference on Machine learning*, page 10961103, 2008.

[18] A. Coates, H. Lee, and A. Y. Ng. An analysis of single-layer networks in unsupervised feature learning. *AISTATS*, 14, 2011.

[19] Eugenio Culurciello, Jordan Bates, Aysegul Dundar, Jose Carrasco, and Clement Farabet. Clustering learning for robotic vision. *International conference on Learning Representations, ICLR 2013 and arXiv:1301.2820*, 2013.

[20] Andrew Saxe, Pang Wei Koh, Zhenghao Chen, Maneesh Bhand, Bipin Suresh, and Andrew Ng. On random weights and unsupervised feature learning. In *International Conference on Machine Learning*, 2011.

[21] Timothee Masquelier, Thomas Serre, Simon Thorpe, and Tomaso Poggio. Learning complex cell invariance from natural videos: A plausibility proof. Technical report, DTIC Document, 2007.

[22] Adam Coates and Andrew Ng. Learning feature representations with k-means. *Neural Networks: Tricks of the Trade*, pages 561–580, 2012.

[23] Y. Boureau, N Le Roux, F. Bach, J. Ponde, and Y. LeCun. Ask the locals: multi-way local pooling for image recognition. In *Proc. International Conference on Computer Vision (ICCV'11)*, 2011.

[24] R. Collobert, K. Kavukcuoglu, and C. Farabet. Torch7: A matlab-like environment for machine learning. In *NIPS Workshop BigLearn*, 2011.

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

[26] Y. Netzer, T. Wang, A. Coates, A. Bissacco, B. Wu, and A. Y. Ng. Reading digits in natural images with unsupervised feature learning. In *NIPS Workshop on Deep Learning and Unsupervised Feature Learning*, volume 2011, 2011.

[27] Ilan Lampl, David Ferster, Tomaso Poggio, and Maximilian Riesenhuber. Intracellular measurements of spatial integration and the max operation in complex cells of the cat primary visual cortex. *Journal of neurophysiology*, 92(5):2704–2713, 2004.

[28] Michael W Spratling. Learning viewpoint invariant perceptual representations from cluttered images. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 27(5):753–761, 2005.

[29] Laurenz Wiskott and Terrence J Sejnowski. Slow feature analysis: Unsupervised learning of invariances. *Neural computation*, 14(4):715–770, 2002.

[30] Guy Wallis, Edmund T Rolls, et al. Invariant face and object recognition in the visual system. *Progress in neurobiology*, 51(2):167–194, 1997.