DE GRUYTER OPEN

JAISCR, 2014, Vol. 4, No. 1, pp. 5 – 19
DOI 10.2478/jaiscr-2014-0021

THE RECOGNITION OF PARTIALLY OCCLUDED OBJECTS WITH SUPPORT VECTOR MACHINES, CONVOLUTIONAL NEURAL NETWORKS AND DEEP BELIEF NETWORKS

Joseph Lin Chu and Adam Krzyżak
Department of Computer Science, Concordia University, Montreal, Quebec, Canada
jo_chu@encs.concordia.ca, krzyzak@cs.concordia.ca

Abstract

Biologically inspired artificial neural networks have been widely used for machine learning tasks such as object recognition. Deep architectures, such as the Convolutional Neural Network, and the Deep Belief Network have recently been implemented successfully for object recognition tasks. We conduct experiments to test the hypothesis that certain primarily generative models such as the Deep Belief Network should perform better on the occluded object recognition task than purely discriminative models such as Convolutional Neural Networks and Support Vector Machines. When the generative models are run in a partially discriminative manner, the data does not support the hypothesis. It is also found that the implementation of Gaussian visible units in a Deep Belief Network trained on occluded image data allows it to also learn to effectively classify non-occluded images. [1]

1 Introduction

Historically, partially occluded object recognition has been rather a challenging task. Most methods of solving this problem have relied on complex preprocessing and feature extraction algorithms, often involving image segmentation and other extra processing [23] [25] [24] [18]. More recent techniques have proposed the use of generative model reconstructions [20].

Convolutional Neural Networks (CNNs) are feed-forward Artificial Neural Networks (ANNs), while Deep Belief Networks (DBNs) make use of of Restricted Boltzmann Machines (RBMs) that use recurrent connections. These networks differ fundamentally in that the DBN is capable of functioning as a generative model, whereas a CNN is merely a discriminative model. A generative model is able to model all variables probabilistically and therefore to generate values for any of these variables. In that sense it can do things like reproduce samples of the original input. A discriminative model on the other hand models only the dependence of an unobserved variable on an observed variable, which is sufficient to perform classification or prediction tasks, but which cannot reproduce samples as a generative model can. This seems to imply that DBNs ought to perform better on the task of partially occluded object recognition, as it should be possible to make use of their generative effects to partially reconstruct the image to aid in classification. We wish to test this hypothesis in our work comparing CNNs, and DBNs.

The preliminary results of these experiments were previously published at the 13th International Conference on Artificial Intelligence and Soft Computing 2014 (ICAISC2014)[4]. This paper extends and details further our research efforts, provid-

[1] This research was supported by the Natural Sciences and Engineering Research Council of Canada

ing additional information, analysis, and considerations.

Section 2 describes in greater detail the learning algorithms used these experiments. Section 3 details our methodology. Section 4 provides an analysis of the experimental results, as well as a more detailed comparison. In Section 5, we discuss these results. And in Section 6 we provide our conclusions as well as consider future research possibilities.

2 Learning Algorithms

In order to best contrast the effectiveness of generative models with discriminative models on the occluded object recognition task, we decided to compare several different machine learning algoirthms: the SVM, the CNN, (two discriminative models) and the DBN, (one generative model). Although the SVM is not really a proper ANN strictly speaking, its considerable popularity as a discriminative classifier meant that it merits inclusion as a kind of control.

2.1 Support Vector Machine

The well-worn SVM is a powerful discriminant classifier that was first developed by Cortes & Vapnik [6]. Although not considered to be an ANN proper, Collobert & Bengio [5] were able to show that they possessed a number of similarities to Perceptrons with the obvious exception of the actual learning algorithm. In addition, CNNs were found to be excellent feature extractors for classifiers such as SVMs as seen in Huang & LeCun [14], as well as Ranzato et al. [21]. This generally involves taking the output of the lower layers of the CNN as feature extractors for the classifier. SVMs are often used with such feature extractors, but can also be trained on raw data, as we choose to do.

2.2 Convolutional Neural Networks

Among the earliest of the hierarchical ANNs based on the visual cortex's architecture was the Neocognitron, first proposed by Fukushima & Miyake [8]. This network was based on the work of neuroscientists Hubel & Wiesel [15], who famously showed the existence of Simple and Complex Cells in the visual cortex. A Simple Cell responds to excitation and inhibition in a specific region of the visual field. A Complex Cell responds to patterns of excitation and inhibition in anywhere a larger receptive field. Together these cells effectively process visual information and perform a delocalization of features in the visual receptive field. Fukushima took the notion of these Simple and Complex Cells to create his Neocognitron, which attempted to implement layers of the artificial equivalent of such neurons in a hierarchical architecture [7]. However, while promising in theory, the Neocognitron had difficulty being implemented effectively, in part because it was originally proposed in the 1980s when computers simply weren't as fast as they are today.

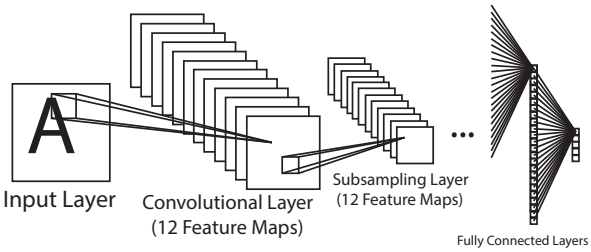

Figure 1. The basic architecture of the CNN.

Then while working at AT&T labs, LeCun et al [16] developed the more practical cnn, which made use of multiple Convolutional and Subsampling layers, while also implementing stochastic gradient descent and backpropagation to construct a feed-forward ANN that performed exceptionally well on image recognition tasks such as the MNIST, which consisted of digit characters. The Convolutional Layer of the CNN is equivalent to the Simple Cell Layer of the Neocognitron, while the Subsampling Layer of the CNN is equivalent to the Complex Cell Layer of the Neocognitron. In essence, these layers function to delocalize features from the visual receptive field, allowing such features to be identified with a degree of shift invariance. This unique structure also allows the CNN to have two important advantages over a fully-connected ANN. First, is the use of the local receptive field, and second is weight-sharing. Both of these advantages have the effect of decreasing the number of weight parameters in the network, thereby making computation of these networks significantly easier than equivalent fully connected networks.

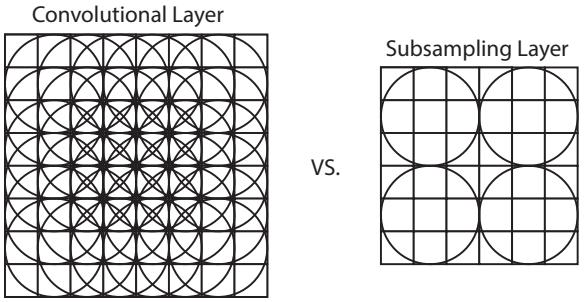

Figure 2. A comparison between the Convolutional layer and the Subsampling layer. Circles represent the receptive fields of the cells of the layer subsequent to the one represented by the square lattice. On the left, an 8 x 8 input layer feeds into a 6 x 6 convolutional layer using receptive fields of size 3 x 3 with an offset of 1 cell. On the right, a 6 x 6 input layer feeds into a 2 x 2 subsampling layer using receptive fields of size 3 x 3 with an offset of 3 cells.

2.3 Deep Belief Networks

The dbn, is one of the more recent developments in machine learning research. The DBN is a recurrent ANN with undirected, bidirectional connections. Structurally, it is made up of multiple layers of RBMs, such that it can be seen as a 'deep' architecture. To understand how this is an effective structure, we must first understand the basic nature of a recurrent ANN.

Recurrent ANNs differ from feed-forward ANNs in that their connections can form cycles. Such networks cannot use feed-forward based learning algorithms, but rather tend to have their own variety of learning algorithms. The advantage of recurrent ANNs is that they can possess associative memory-like behaviour. Early recurrent ANNs, such as the Hopfield network [13], showed promise in this regard, but suffered certain limitations. The Hopfield network was only a single layer architecture that was only able to learn very limited problems due to limited memory capacity. A multi-layer generalization of the Hopfield Network was developed, known as the Boltzmann Machine [1], which while able to store considerably more memory suffered from being overly slow to train. Nevertheless, the groundwork was laid for more sophisticated developments.

A Boltzmann Machine is considered an energy-based model. This means that it has a scalar energy that represents a particular configuration of variables. Such an energy-based model learns by changing its energy function such that it has a shape that possesses desirable properties. Commonly this corresponds to having a low energy. Thus the algorithm performs learning by trying to find a way to minimize the energy of the Boltzmann Machine. The energy of a Boltzmann Machine can be defined by (1):

$$E(x,h) = \\ =\sum_{i\in visible} a_i x_i - \sum_{j\in hidden} b_j h_j - \sum_{i,j} x_i h_j w_{i,j} - \sum_{i,j} x_i x_j u_{i,j} - \sum_{i,j} h_i h_j v_{i,j} \tag{1}$$

This in turn is applied to a probability distribution:

$$P(x) = \frac{e^{-E(x)}}{Z} \tag{2}$$

where Z is the partition function:

$$Z = \sum_x e^{-E(x)} \tag{3}$$

These equations we modify to incorporate hidden variables:

$$P(x,h) = \frac{e^{-E(x,h)}}{Z} \tag{4}$$

$$P(x) = \sum_h P(x,h) \\ = \sum_h \frac{e^{-E(x,h)}}{Z} \tag{5}$$

$$Z = \sum_x e^{-E(x,h)} \tag{6}$$

The concept of Free Energy is borrowed from physics, where it is the useable subset of energy, and represents a marginalization of the energy in the log domain:

$$F(x) = -log\sum_{h} e^{-E(x,h)} \tag{7}$$

Let θ represent the parameters of the model. The data log-likelihood gradient thus becomes:

$$\begin{aligned}\frac{\partial logP(x)}{\partial\theta} &= -\frac{\partial F(x)}{\partial\theta} + \frac{1}{Z}\sum_{\tilde{x}} e^{-F(\tilde{x})}\frac{\partial F(x)}{\partial\theta} \\ &= -\frac{\partial F(x)}{\partial\theta} + \sum_{\tilde{x}} P(\tilde{x})\frac{\partial F(x)}{\partial\theta}\end{aligned} \tag{8}$$

Possession of this gradient enables us to perform a kind of stochastic gradient descent as a means of finding that desired lowest energy state mentioned earlier. However, in practice, this gradient is difficult to calculate for a regular Boltzmann Machine, and while not intractable, it is a very slow computation. Thus, a way to accelerate the process was sought.

A variant of the Boltzmann Machine was first known as a Harmonium [22], but later called a rbm, which initially saw little use. Then a fast learning algorithm for RBMs called Contrastive Divergence was developed by Hinton [9]. Contrastive Divergence uses Gibbs sampling within a gradient descent process. The rbm itself primarily differs from a regular general Boltzmann Machine by the simple fact that it lacks the lateral or sideways connections within layers. As such an rbm can be defined by the much simpler energy function as follows:

$$E(v,h) = \sum_{i\in visible} a_i v_i - \sum_{j\in hidden} b_j h_j - \sum_{i,j} v_i h_j w_{i,j} \tag{9}$$

where v_i and h_j are the binary states of the visible unit i and hidden unit j, a_i and b_j are their biases, and $w_{i,j}$ is the weight connection between them [10].

Applying the data log-likelihood gradient from earlier, we can now find the derivative of the log probability of a training vector with respect to a weight:

$$\frac{\partial logp(v)}{\partial w_{ij}} = \langle v_i h_j\rangle_{data} - \langle v_i h_j\rangle_{model} \tag{10}$$

where, the angle brackets enclose the expectations of the distribution labeled in the subscript. And thus, the change in a weight in an rbm is given by the learning rule:

$$\Delta w_{i,j} = \varepsilon(\langle v_i h_j\rangle_{data} - \langle v_i h_j\rangle_{model}) \tag{11}$$

where ε is the learning rate.

$\langle v_i h_j\rangle_{data}$ is fairly easy to calculate. If you take a randomly selected training vector v, then the binary state h_j of each of the hidden units is 1 with probability:

$$p(h_j = 1|v) = \sigma(b_j + \sum_i v_i w_{ij}) \tag{12}$$

where $\sigma(x)$ a logistic sigmoid function such as $1/(1+exp(-x))$.

Similarly, given a hidden vector, we can get an unbiased sample of the state of a visible unit:

$$p(v_j = 1|h) = \sigma(a_i + \sum_j h_j w_{ij}) \tag{13}$$

$\langle v_i h_j\rangle_{model}$ is much more difficult to calculate, and so we use an approximation $\langle v_i h_j\rangle_{recon}$ instead. Basically this reconstruction requires first setting the visible units to a training vector, and then computing the binary states of the hidden units in parallel with equation 12. Next, set each v_i to 1 with a probability according to equation 13, and we get a reconstruction.

$$\Delta w_{i,j} = \varepsilon(\langle v_i h_j\rangle_{data} - \langle v_i h_j\rangle_{recon}) \tag{14}$$

This learning rule attempts to approximate the gradient of an objective function called the Contrastive Divergence (which itself is an approximation of the log-likelihood gradient), though it is not actually following the gradient exactly. Despite this approximation, the rule works quite well for many applications, and is much faster than the previously mentioned way of learning regular general Boltzmann Machines.

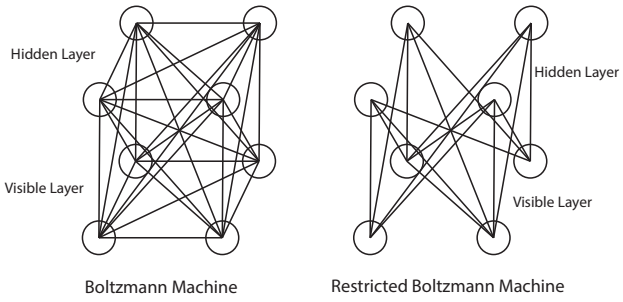

Figure 3. The structure of the general Boltzmann Machine and the rbm.

The DBN is created by stacking RBMs together, as Hinton, Osindero, & Teh, [11] discovered. This DBN is then trained in a greedy, layer-wise fashion. In general, this involves pre-training each RBM separately, starting at the bottom layer and working gradually up to the top layer. All layers thus have their weights initialized using unsupervised learning in the pre-training phase, after which fine-tuning using Backpropagation is performed using the labeled data, training in a supervised manner.

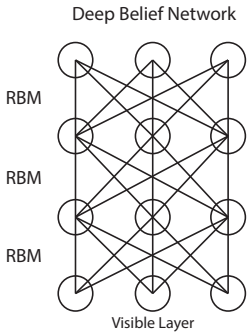

Figure 4. The structure of the DBN.

Mathematically, we can describe a DBN with l layers according to the joint distribution below, given an observed vector x, and l hidden layers h^k [2].

$$P(x,h^1,...,h^l) = \left(\prod_{k=0}^{l-2} P(h^k|h^{k+1})\right) P(h^{l-1},h^l) \tag{15}$$

In this case, $x = h^0$, while $P(h^{k-1}|h^k)$ is a visible-given-hidden conditional distribution in the RBM at level k of the DBN, and $P(h^{l-1},h^l)$ is the top-level RBM's joint distribution.

The DBN, when introduced, produced then state of the art performance on such tasks as the MNIST. Later, DBNs were also applied to 3D object recognition [19]. Ranzato, Susskind, Mnih, & Hinton [20] also showed how effective DBNs could be on occluded facial images.

3 Methodology

The object/image dataset we used was the small NORB [17]. The small NORB consists of 5 object categories and several thousand images per category, for a total of 24300 images each in the training and test sets. The small NORB proper technically includes a pair of stereo images for each training example, but for our purposes, we chose to only use one of the images in the pair. Normal, non-occluded images with the object fully visible in the image are seen in figure 5. Occluded images were created by occluding a random half of each image in the test set with zeroes (black) as shown in figure 6.

Figure 5. Images from the small NORB non-occluded data set.

For the SVMs we tested various parameters from the literature, such as Huang & LeCun [14] and Ranzato et al. [21] and eventually settled on a Gamma value of 0.0005, and a C value of 40. Gamma is how far a single training example affects things, with low values being "far" and high values being "close". C is the tradeoff between misclassifying as few training samples as possible (high C) and a smooth decision surface (low C). For code for the SVMs, we used the library "LIBSVM" by Chih-

Chung Chang and Chih-Jen Lin from the National Taiwan University [3].

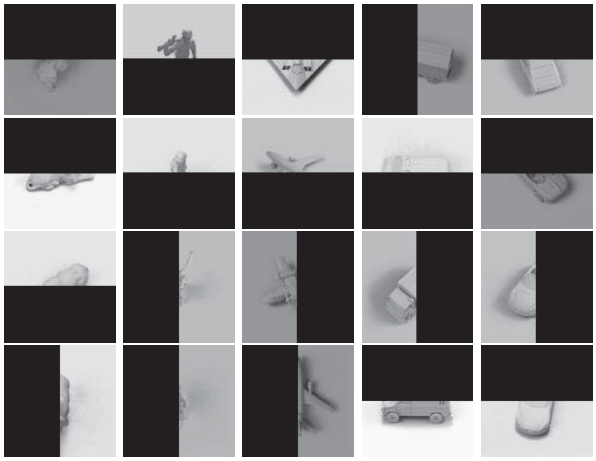

Figure 6. Images from the small NORB occluded data set.

For the CNN, Sirotenko's Matlab library "CNN – Convolutional neural network class" (http://www.mathworks.com/matlabcentral/fileexchange/24291-cnn-convolutional-neural-network-class) was modified extensively to serve our purposes. CNNs require very particular considerations when implementing their architecture, as there are a sizable number of parameters to describe each layer. A method was devised to calculate a usable set of architecture parameters. The relationship between layers can be described as follows. To calculate the reasonable dimensions of a square layer from either its previous layer (or next layer) in the hierarchy requires at least some of the following variables to be assigned. Let x be the width of the previous (or current) square layer. Let y be the width of the current (or next) square layer. Let r be the width of the square receptive field of nodes in the previous (or current) layer to each current (or next) layer node, and f be the offset distance between the receptive fields of adjacent nodes in the current (or next) layer. The relationship between these variables is best described by the Equation 16.

$$y = \frac{x-(r-f)}{f} \tag{16}$$

where, $x \geq y$, $x \geq r \geq f$, and $f > 0$

For convolutional layers this generalizes because f = 1, to equation 17.

$$y = x - r + 1 \tag{17}$$

For subsampling layers, this generalizes because r = f, to equation 18.

$$y = \frac{x}{f} \tag{18}$$

From this we can determine the dimensions of each layer. The architecture for the CNN on the NORB dataset is shown in Table 1, where S, C and F represent convolutional, subsampling and fully connected layers, respectively.

Table 1. The architecture of the CNN used on the NORB dataset, based on Huang & LeCun [14].

CNN			
Layer	**Nodes**	**k or r**	**Feature Maps**
S1	96x96		
C2	92x92	5	8
S3	23x23	4	8
C4	18x18	6	24
S5	6x6	3	24
C6	1x1	6	24
F1	100	1	
F2	5	1	

For the CNN, various other parameters were also experimented with to determine the optimal parameters to use in our experiments. We eventually settled on 100 epochs of training. The CNN learning rate and learning rate decrement parameters were determined by using Huang and LeCun's recommendations [14]. That is to say, the learning rate was initially set to 2.00E-05, and gradually decremented to approximately 2.00E-07.

For the DBN we used Stansbury's Matlab library "Matlab Environment for Deep Architecture Learning (MEDAL)" (https://github.com/dustinstansbury/medal). Experiments were also conducted on the parameters for the DBN. DBNs use binary visible units by default. However, a modification has been recommended to implement Gaussian visible units for image data [12]. Thus, DBNs using both binary and Gaussian visible units were tested.

Table 2. The parameters chosen as an optimal configuration for the DBNs.

Parameters - DBN						
Parameters			Learning Rate		Epochs	
Visible	Layers	Hidden	Pre-Training	Fine-Tune	Pre-Training	Fine-Tune
Binary	2	2000	0.1	0.01	200	50
Binary	2	4000	0.1	0.01	200	50
Gaussian	2	4000	0.001	0.001	200	50

In addition, two different quantities of hidden nodes were used, 2000 and 4000 respectively for the binary visible unit based DBNs. This was because prior experiments used to determine the effectiveness of various parameter configurations found unexpectedly that the binary units in combination with 2000 hidden nodes seemed to actually perform better than the combination of binary units and 4000 hidden nodes. Gaussian visible units on the other hand, showed a greater degree of effectiveness at 4000 hidden nodes, than at 2000 hidden nodes, which was more within expectation. We tested multiple configurations for these reasons. Eventually, through systematic efforts involving testing various parameters at different values and looking at the change in performance, we settled on the Layer, Learning Rate, and Epoch parameters for the Visible and Hidden Node cases shown in Table 2. Hinton's guide also provided some suggested values that we took into consideration [10].

In Table 2, Visible indicates the type of visible unit used in the input layer. Layers indicates the number of RBMs in the DBN. Hidden indicates the number of hidden nodes per layer. Learning Rate is divided between the learning rate for the DBN during initial pre-training, and during fine-tuning using Backpropagation. Epochs is similarly divided. Epochs indicates the number of times the network was trained on the training data.

Some additional parameters we settled on are shown in Table 3, some of which were based on experimentation, while others were simply default settings that worked well. Momentum indicates the parameter for momentum as described by Hinton [10]. Weight Penalty indicates the value by which weights were penalized by during training. Batch Size indicates the number of samples in each batch fed to the network during learning. "Begin Anneal At" indicates at what Epoch Simulated Annealing is started. "Number of Gibbs Sampling" indicates how many times during Contrastive Divergence the Gibbs sampling process was run. Sparsity is a parameter that controls the degree to which sparsity affects the network. "Start to Vary Eta At" indicates at what Epoch the learning rate starts to be varied. "Display Every" is a parameter for indicating how often information is displayed during the training process.

Finally, experiments were performed with the optimized parameters for the SVMs, CNNs, and DBNs on the small NORB image dataset. Each of the training and testing sets consisted of 24300 images. These experiments consisted of three different methods of training: one which consisted of training exclusively on the non-occluded training set, followed by testing on both a non-occluded test set and an occluded test set; one which consisted of training exclusively on the occluded training set, followed by testing on both a non-occluded test set and an occluded test set; and finally one which consisted of training on a mixture of non-occluded and occluded images, followed by testing on both a non-occluded test set and an occluded test set. Three replicates were performed for each experimental setup and averaged.

Table 3. The parameters chosen as an optimal configuration for the DBNs.

Parameters - DBN	
Momentum	0.5
Weight Penalty	2.00E-04
Batch Size	100
Begin Anneal At	50
Number of Gibbs Sampling	1
Sparsity	0.01
Start to Vary Eta At	50
Display Every	20

4 Analysis and Results

4.1 Support Vector Machines

Table 4 provides a direct comparison of the non-occluded, occluded, and mixed trained SVMs. The results show that when the SVM is trained with only non-occluded training images, its object recognition performance on non-occluded test images is reasonably good (83% accuracy), but the same algorithm performs poorly on the occluded test images (20% accuracy). On the other hand, when the SVM is trained with only occluded training images, it performs reasonably well at object recognition on the occluded test images (69% accuracy), but performs poorly on the non-occluded test images (20% accuracy).

The results also showed that training the SVM on the mixed data set containing both non-occluded and occluded images, led it to do well at object recognition on the non-occluded test images (81% accuracy), on the occluded test images (69% accuracy), and also on the mixed test images (75% accuracy). These accuracies on the non-occluded test images are comparable to the SVMs trained on the non-occluded training images (i.e., 81% vs. 83%). Furthermore, its accuracy on the occluded test images is comparable to the SVMs trained on the occluded training images (i.e., 69% vs. 69%).

When comparing performance on the mixed test data set, the SVM performed better if trained with the mixed training data set (75% accuracy), than if it was trained using the non-occluded training images (51% accuracy), or the occluded training images (45% accuracy) alone.

When testing the SVM on the same training data set as it was trained on, the accuracy on the object recognition task was slightly lower when using the mixed training images (97% accuracy) than if it was trained exclusively on the non-occluded training images (99.9% accuracy), or exclusively on the occluded training images (99% accuracy). To understand this, the concept of overfitting must be considered. Overfitting occurs when a learning algorithm learns to fit itself to the training data, rather than learning the general concept that it is desired to learn from the training data. In such an instance, the performance on the test data can actually decrease while accuracy on the original training data continues to increase. Obviously, performance on the original training data should be better than performance on previously unseen test data, but it is possible that if the difference between the two performances is significant, this could indicate some degree of overfitting.

4.2 Convolutional Neural Networks

Table 5 provides a direct comparison of the non-occluded, occluded, and mixed trained CNNs. The results show that when the CNN is trained with only non-occluded training images, its object recognition performance on non-occluded test images is reasonably good (83% accuracy), but the same algorithm performs poorly on the occluded test images (20% accuracy). On the other hand, when the CNN is trained with only occluded training images, it performs reasonably well at object recognition on the occluded test images (59% accuracy), but performs poorly on the non-occluded test images (30% accuracy), albeit notably better than the equivalent SVM.

The results also showed that training the CNN on the mixed data set containing both non-occluded and occluded images, led it to do well at object recognition on the non-occluded test images (77% accuracy), on the occluded test images (67% accuracy), and also on the mixed test images (72% accuracy). This accuracy on the non-occluded test images is comparable to the CNNs trained on the non-occluded training images (i.e., 77% vs. 83%). Furthermore, the mixed trained CNN's accuracy on the occluded test images is notably better than the CNNs trained on the occluded training images (i.e., 67% vs. 59%).

When comparing performance on the mixed test data set, the CNN performed better if trained with the mixed training data set (72% accuracy), than if trained with the occluded training images (44% accuracy) alone.

When testing the CNN on the same training data set as it was trained on, the accuracy on the object recognition task was significantly lower when using the mixed training images (83% accuracy) than if it was trained exclusively on the non-occluded training images (96% accuracy). Testing on the occluded training images actually had the lowest performance of the three options (69% accuracy). It is possible that this suggests that there was less over-

Table 4. A comparison of the accuracy results of the non-occluded, occluded, and mixed trained SVMs.

SVM - NORB				
Training	Training Test	Mixed Test	Non-Occluded Test	Occluded Test
Non-Occluded	**0.999 ± 0.003**	0.513 ± 0.001	**0.825 ± 0.007**	0.200 ± 0.003
Occluded	0.994 ± 0.0001	0.446 ± 0.0002	0.200 ± 0.0001	0.692 ± 0.0005
Mixed	0.973 ± 0.0003	**0.754 ± 0.001**	0.813 ± 0.001	**0.694 ± 0.0005**

Note: Mean of 3 replicates ± standard error.

Table 5. A comparison of the accuracy results of the non-occluded, occluded, and mixed trained CNNs.

CNN - NORB				
Training	Training Test	Mixed Test	Non-Occluded Test	Occluded Test
Non-Occluded	**0.955 ± 0.000**	0.515 ± 0.000	**0.831 ± 0.000**	0.199 ± 0.000
Occluded	0.693 ± 0.003	0.444 ± 0.017	0.304 ± 0.031	0.585 ± 0.002
Mixed	0.832 ± 0.002	**0.717 ± 0.003**	0.769 ± 0.009	**0.665 ± 0.010**

Note: Mean of 3 replicates ± standard error.

fitting on mixed and occluded training images than on the non-occluded training images.

4.3 Deep Belief Networks

Tables 6-8 provide a direct comparison of the non-occluded, occluded, and mixed trained DBNs, with the differences between each table resulting from the effects of choosing different visible units and number of hidden units in the ANN. An ANN is generally divided into layers, with the first layer being the input or visible layer containing visible units, while the last layer is the output layer. In between these two layers, can be any number of hidden layers containing hidden nodes. For the purposes of experimentation, two different types of visible units, binary and Gaussian, were used, while two different amounts of hidden nodes were used as well, 2000 and 4000 respectively for the binary units. This was because prior experiments used to determine the effectiveness of various parameter configurations found that the binary units in combination with 2000 hidden nodes seemed to actually perform better than the combination of binary units and 4000 hidden nodes, which was different than expected. Gaussian units on the other hand, showed greater effectiveness at 4000 hidden nodes, than at 2000 hidden nodes, which was expected. For this reason, we tested multiple configurations as shown.

For simplicity, we can refer to these configurations as B-2000 for binary visible units and 2000 hidden nodes, B-4000 for binary visible units and 4000 hidden nodes, and G-4000 for Gaussian visible units and 4000 hidden nodes.

Table 6 shows specifically the performance of the DBNs using binary visible units and having 2000 hidden nodes. As expected, the DBN trained on the non-occluded training images achieves a respectable performance (87% accuracy) on the non-occluded test images, while not performing so well on the occluded test images (21% accuracy). Conversely, the DBN trained on the occluded training images managed to achieve reasonably good results on the occluded test images (71% accuracy), while not fairing so well on the non-occluded test images (19% accuracy).

The DBN trained on the mixed training images managed a somewhat lower performance on the non-occluded test set than the DBN trained exclusively on the non-occluded training images (68% vs. 87% accuracy), and a slightly lower performance on the occluded test set than the DBN trained exclusively on the occluded training images (68% vs. 71% accuracy).

When comparing performance on the mixed test data set, the DBN performed better if trained with the mixed training data set (68% accuracy), than if it was trained using the occluded training images (45% accuracy) alone.

When testing the DBN on the same training data set as it was trained on, the accuracy on the object recognition task was significantly lower when using the mixed training images (83% accuracy) or the

Table 6. A comparison of the accuracy results of the non-occluded, occluded, and mixed trained DBNs using binary visible units with 2000 hidden nodes.

DBN - Binary Visible Unit w/ 2000 Hidden Nodes				
Training	Training Test	Mixed Test	Non-Occluded Test	Occluded Test
Non-Occluded	**0.993 ± 0.0002**	0.545 ± 0.000	**0.873 ± 0.007**	0.214 ± 0.004
Occluded	0.847 ± 0.007	0.451 ± 0.026	0.193 ± 0.044	**0.708 ± 0.009**
Mixed	0.832 ± 0.013	**0.680 ± 0.013**	0.676 ± 0.037	0.684 ± 0.020

Note: Mean of 3 replicates ± standard error.

Table 7. A comparison of the accuracy results of the non-occluded, occluded, and mixed trained DBNs using binary visible units with 4000 hidden nodes.

DBN - Binary Visible Unit w/ 4000 Hidden Nodes				
Training	Training Test	Mixed Test	Non-Occluded Test	Occluded Test
Non-Occluded	**0.989 ± 0.002**	0.520 ± 0.008	**0.841 ± 0.014**	0.203 ± 0.002
Occluded	0.852 ± 0.007	0.458 ± 0.014	0.208 ± 0.022	**0.708 ± 0.006**
Mixed	0.866 ± 0.008	**0.673 ± 0.001**	0.653 ± 0.004	0.693 ± 0.004

Note: Mean of 3 replicates ± standard error.

Table 8. A comparison of the accuracy results of the non-occluded, occluded, and mixed trained DBNs using Gaussian visible units with 4000 hidden nodes.

DBN - Gaussian Visible Unit w/ 4000 Hidden Nodes				
Training	Training Test	Mixed Test	Non-Occluded Test	Occluded Test
Non-Occluded	**0.981 ± 0.003**	0.550 ± 0.002	**0.832 ± 0.006**	0.258 ± 0.013
Occluded	0.786 ± 0.001	0.673 ± 0.002	0.693 ± 0.006	0.652 ± 0.005
Mixed	0.860 ± 0.016	**0.697 ± 0.023**	0.714 ± 0.044	**0.679 ± 0.006**

Note: Mean of 3 replicates ± standard error.

Table 9. Comparison of the accuracy results of the Classifier Algorithms on the Non-Occluded Training Images

Trained On Non-Occluded Training Image Set				
Classifier	Training Test	Mixed Test	Non-Occluded Test	Occluded Test
SVM	**0.999 ± 0.003**	0.513 ± 0.001	0.825 ± 0.007	0.200 ± 0.003
CNN	0.955 ± 0.000	0.515 ± 0.000	0.831 ± 0.000	0.199 ± 0.000
DBN (B-2000)	0.993 ± 0.0002	0.545 ± 0.000	**0.873 ± 0.007**	0.214 ± 0.004
DBN (B-4000)	0.989 ± 0.002	0.520 ± 0.008	0.841 ± 0.014	0.203 ± 0.002
DBN (G-4000)	0.981 ± 0.003	**0.550 ± 0.002**	0.832 ± 0.006	**0.258 ± 0.013**

Note: Mean ± standard error.

occluded training images (85% accuracy), than if it was trained exclusively on the non-occluded training images (99% accuracy).

Table 7 shows specifically the performance of the DBNs using binary visible units and having 4000 hidden nodes. As expected, the DBN trained on the non-occluded training images achieves a respectable performance (84% accuracy) on the non-occluded test images, while not performing so well on the occluded test images (20% accuracy). Conversely, the DBN trained on the occluded training images managed to achieve reasonably good results on the occluded test images (71% accuracy), while not fairing so well on the non-occluded test images (21% accuracy).

The DBN trained on the mixed training images managed a somewhat lower performance on the non-occluded test set than the DBN trained exclusively on the non-occluded training images (65% vs. 84% accuracy), and a slightly lower performance on the occluded test set than the DBN trained exclusively on the occluded training images (69% vs. 71% accuracy).

When comparing performance on the mixed test data set, the DBN performed better if trained with the mixed training data set (67% accuracy), than if it was trained using the occluded training images (46% accuracy) alone.

When testing the DBN on the same training data set as it was trained on, the accuracy on the object recognition task was significantly lower when using the mixed training images (87% accuracy) or the occluded training images (85% accuracy), than if it was trained exclusively on the non-occluded training images (99% accuracy).

Table 8 shows specifically the performance of the DBNs using Gaussian visible units and having 4000 hidden nodes. As expected, the DBN trained on the non-occluded training images achieves a respectable performance (83% accuracy) on the non-occluded test images, while not performing so well on the occluded test images (26% accuracy). Conversely, the DBN trained on the occluded training images managed to achieve reasonably good results on the occluded test images (65% accuracy), while also doing quite well on the non-occluded test images (69% accuracy). This discovery of better performance when trained with occluded images and tested with non-occluded images in G-4000 could be due to the generative model's ability to learn to classify whole images using features learned from the partial images of the occluded images.

The DBN trained on the mixed training images managed a somewhat lower performance on the non-occluded test set than the DBN trained exclusively on the non-occluded training images (71% vs. 83% accuracy), and a slightly higher performance on the occluded test set than the DBN trained exclusively on the occluded training images (68% vs. 65% accuracy). This superior performance by the mixed trained SVM on the occluded test set was unexpected, and could be due to the larger size of the mixed training data set, which essentially includes all the images from the non-occluded training data set, and all the images from the occluded training data set.

When comparing performance on the mixed test data set, the DBN performed better if trained with the mixed training data set (70% accuracy), than if it was trained using the occluded training images (67% accuracy) alone.

When testing the DBN on the same training data set as it was trained on, the accuracy on the object recognition task was significantly lower when using the mixed training images (86% accuracy) or the occluded training images (79% accuracy), than if it was trained exclusively on the non-occluded training images (98% accuracy).

4.4 Comparison

Table 9 compares the various classification algorithms that have been trained on the non-occluded images. As expected, the performance on the non-occluded test images is reasonably high (83-87% accuracy). Conversely performance on the occluded test images is expectedly poor (20-26% accuracy).

Performance on the mixed test images appears to be the average of the non-occluded and occluded test image performances (51-55% accuracy).

Performance on the original training images is very high across the board (96-99% accuracy).

Table 10 compares the various classification algorithms that have been trained on the occluded images. While the SVM, DBN (B-2000), and the

DBN (B-4000) achieve around chance on the non-occluded test images (19-21% accuracy), the CNN achieves a somewhat higher than chance result (30% accuracy), and the DBN (G-4000) achieves a remarkably high result (69% accuracy). These results are unusual because one would expect that classifiers trained on the occluded test set exclusively might perform at chance on the non-occluded test set (20% accuracy). However, several of the algorithms used achieved significantly higher numbers on a test set type that it wasn't trained on. This suggests that learning the occluded set is actually sometimes useful to classifying on the non-occluded set. Meanwhile, the performance on the occluded test images is closer to expected (59-71% accuracy).

Performance on the mixed test images appears to be the average of the non-occluded and occluded test image performances (44-67% accuracy).

Performance on the original training images is more varied than with the non-occluded (69-99% accuracy), though still consistently higher than performance on the test images.

Table 11 compares the various classification algorithms that have been trained on the mixed image set. As expected, the performance on the non-occluded test images is reasonable (65-81% accuracy), albeit within a significantly wider range than the performance on the occluded test images (67-69% accuracy).

Performance on the mixed test images appears to be the average of the non-occluded and occluded test image performances (67-75% accuracy).

Performance on the original training images (83-97% accuracy) is lower than with the non-occluded images though less varied than the occluded images, and it remains consistently higher than performance on the test images.

5 Discussion

Our experiments appear to show that when training a classifier on only the non-occluded training set, the occluded task is a particularly challenging one for both discriminative models, such as SVMs and CNNs, and generative models, namely the DBNs. In general, training on the non-occluded images tends to lead to good performance on the non-occluded test set, but poor performance on the occluded test set, while in most cases, training on the occluded images leads to good performance on the occluded test set, and poorer performance on the non-occluded test set.

However, it appears that training on the occluded training set only, for DBNs using Gaussian visible units at least, produces a highly unusual result of good performance on the non-occluded test set (69% accuracy). This behaviour is not readily apparent with the DBN using binary visible units (19-21% accuracy). A much less pronounced but similar effect is also visible with the CNN (30% accuracy), which is not seen at all with SVM, which performs at chance (20% accuracy). It may well be that this is because the SVM is a purely discriminative model. The CNN, while also a discriminative model, is also an ANN, which perhaps gives it some greater similarity to the DBN. Nevertheless, the unexpectedly good performance of the Gaussian visible unit based DBN on the dataset type it wasn't trained on is something perhaps worth looking into for future research. Note though that this seems to come at a cost to performance on occluded test set, as it was the only classifier that performed better on the dataset type it wasn't trained on (69% accuracy), than on the type it was trained on (65% accuracy).

Training the SVM, the CNN, and the DBN with Gaussian visible units on a mixed training set containing both non-occluded and occluded images leads to slightly worse performance on the non-occluded test set than an exclusively non-occluded trained classifier, and slightly better performance on the occluded test set than an exclusively occluded trained classifier. These results suggest that mixed training actually improves performance on the occluded problem to an extent. It is potentially possible that these classifiers are benefiting from the more complete images in the non-occluded part of the training set.

Meanwhile, training a DBN with binary visible units on a mixed training set containing both non-occluded and occluded images causes it to perform worse on the non-occluded test set than a pure non-occluded training set, and is worse but is very close in performance on the occluded test set to that trained on a pure occluded training set. This is anticipated, as a mixed training set should yield mediocre performance on both test sets compared to

Table 10. Comparison of the accuracy results of the Classifier Algorithms on the Occluded Training Images

Trained On Occluded Training Image Set				
Classifier	Training Test	Mixed Test	Non-Occluded Test	Occluded Test
SVM	**0.994 ± 0.0001**	0.446 ± 0.0002	0.200 ± 0.0001	0.692 ± 0.0005
CNN	0.693 ± 0.003	0.444 ± 0.017	0.304 ± 0.031	0.585 ± 0.002
DBN (B-2000)	0.847 ± 0.007	0.451 ± 0.026	0.193 ± 0.044	**0.708 ± 0.009**
DBN (B-4000)	0.852 ± 0.007	0.458 ± 0.014	0.208 ± 0.022	**0.708 ± 0.006**
DBN (G-4000)	0.786 ± 0.001	**0.673 ± 0.002**	**0.693 ± 0.006**	0.652 ± 0.005

Note: Mean ± standard error.

Table 11. Comparison of the accuracy results of the Classifier Algorithms on the Mixed Training Images

Trained On Mixed Training Image Set				
Classifier	Training Test	Mixed Test	Non-Occluded Test	Occluded Test
SVM	**0.973 ± 0.0003**	**0.754 ± 0.001**	**0.813 ± 0.001**	**0.694 ± 0.0005**
CNN	0.832 ± 0.002	0.717 ± 0.003	0.769 ± 0.009	0.665 ± 0.010
DBN (B-2000)	0.832 ± 0.013	0.680 ± 0.013	0.676 ± 0.037	0.684 ± 0.020
DBN (B-4000)	0.866 ± 0.008	0.673 ± 0.001	0.653 ± 0.004	0.693 ± 0.004
DBN (G-4000)	0.860 ± 0.016	0.697 ± 0.023	0.714 ± 0.044	0.679 ± 0.006

Note: Mean ± standard error.

classifiers trained exclusively on the non-occluded or the occluded training sets.

While training a SVM, CNN, and a DBN with Gaussian visible units on a mixed training set leads to better relative performance on the non-occluded test image set than on the occluded test image set, it appears that the reverse is the case with DBNs with binary visible units, which had better relative performance on the occluded test image set than on the non-occluded test image set. This is a somewhat curious discovery, and may be indicative of the differences between binary and Gaussian visible units.

When compared to other research work in the literature, the experiments performed on the SVM and CNN did not exceed the performance of the results from Huang and LeCun [14]. Huang and LeCun were able to achieve 88.4% accuracy with their SVM on the small NORB dataset, and 93.8% accuracy with their CNN on the small NORB dataset [14]. The SVM in our experiments, with the same parameters as Huang and LeCun [14], achieved 82.5% ± 0.7% accuracy, while our CNN achieved 83.1% accuracy. Our best performing algorithm was actually a DBN using binary visible units and 2000 hidden nodes, which achieved 87% accuracy. In comparison, Nair and Hinton [19], achieved 93.5% accuracy with their DBN on the standard small NORB dataset, and 94.8% accuracy with their DBN using extra unlabeled data. Thus, on the non-occluded images, we did not quite achieve the best results in the literature.

A significant possible reason for our relatively inferior performance was that we chose to take only one of the two stereo images in the NORB dataset to be used by our learning algorithms. On the other hand, the top performing results in the literature generally made use of both stereo images. We made the decision not to use the stereo pair images primarily because of practical limitations on our part, namely that it would double the size of the dataset in memory, and that in the case of the CNN it would require a considerable amount of extra modification to the architecture of the network. Thus, we chose to save both memory and time by using only the single image. This was an important choice, as we were significantly limited in the amount of RAM available on our computers, and the amount of time required to train with even this limited version of the NORB was rather substantial already. Also, in real life situations, it is often difficult to obtain stereo images without resorting to some special robotic vision setup. Conversely, single images are readily available in many datasets, CCTV cameras, and Internet searches.

In so far as occluded images are concerned, there remains a lack of results in the literature that are directly comparable to our work. What seems to be the the most similar work done so far would be the research of Ranzato et al. [20]. Their work on classifying facial expressions includes some use of occlusion. Rather than making use of NORB, they used the CK dataset, and the TFD, classifying 7 different facial expressions, rather than 5 objects. Their Type 3 - right half, Type 4 - bottom half, and Type 5 - top half occlusions are most similar to the occlusions we used in our experiments. Importantly, their deep generative model actually attempts to reconstruct the image first before classifying, unlike our experiments. This thus takes full advantage of the unique properties of generative models. As such, they achieved fairly impressive results in their experiments.

Overall, the results of Ranzato et al. [20], in combination with our own results, appear to suggest that the advantage of implementing a generative model comes from the reconstruction process that Ranzato et al. [20] were able to use, and is not merely a result of classification using a generative model discriminatively as we did. Further research naturally could involve the actual implementation of some kind of reconstruction process similar to what Ranzato et al. [20] used, except on the small NORB dataset, to determine whether or not this conjecture actually holds.

Furthermore, a possible reason why the performance of the generative DBN did not exceed the discriminative models could be due to the fact that the DBNs were fine-tuned with Backpropagation. As such a process is inherently discriminative rather than generative, the final resulting network may perhaps behave more like a discriminative model than a generative model. If this is the case, we should be able to see some difference in the accuracy of the model when it has only been pre-trained, and not yet fine-tuned with Backpropagation. To truly test this possibility, it may be necessary to run the experiments again using a generative model that is fully generative through and through, such as a DBM.

6 Conclusion

It would therefore appear to us that our original hypothesis that the generative models would perform significantly better on the occluded task than the discriminative models lacks empirical support. Rather, it appears that, when run in a partially discriminative manner, the generative model, in our case the DBN, appears to perform approximately equally well to discriminative models, such as the SVM and the CNN. This strongly implies that, with regards to other findings in the literature which use generative models and do in fact show notable differences, that these differences are primarily due to the implementation of additional reconstruction processes, and is not due merely to the architecture and training algorithm itself.

In addition, with regards to certain DBNs that implement Gaussian visible units, when trained on the occluded training set and tested on the non-occluded dataset, they show a remarkable performance that perhaps warrants additional investigation. Furthermore, these results may suggest that, at least when using this particular variant of DBN, intentionally occluding data sets may allow for good performance on both the non-occluded and occluded tasks. Such techniques could very well prove useful in tasks in which the original training set is non-occluded, but the real-world test data may include occlusions, such as in the case of real-world face recognition from CCTV cameras.

In any case, it appears that all of the tested learning algorithms are potentially competitive methods of performing partially occluded object recognition, as long as the training data is appropriately pre-processed with at least some artificial occlusions.

References

[1] D. H. Ackley, G. E. Hinton, and T. J. Sejnowski. A learning algorithm for boltzmann machines. *Cognitive Science*, 9:147–169, 1985.

[2] Y. Bengio. Learning deep architectures for ai. *Foundations and Trends in Machine Learning*, 2(1):1–127, 2009.

[3] Chih-Chung Chang and Chih-Jen Lin. LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, 2:27:1–27:27, 2011. Software available at http://www.csie.ntu.edu.tw/ cjlin/libsvm.

[4] Joseph Lin Chu and Adam Krzyżak. Application of support vector machines, convolutional neural networks and deep belief networks to recognition of

partially occluded objects. In L. Rutkowski, editor, *The 13th International Conference on Artificial Intelligence and Soft Computing ICAISC 2014, Lecture Notes on Artifical Intelligece (LNAI)*, volume 8467, pages 34–46. Springer International Publishing Switzerland, 2014.

[5] R. Collobert and S. Bengio. Links between perceptrons, mlps and svms. *Proceedings of the 21st International Conference on Machine Learning*, page 23, 2004.

[6] C. Cortes and V. N. Vapnik. Support-vector networks. *Machine Learning*, 20:273–297, 1995.

[7] K. Fukushima. Neocognitron for handwritten digit recognition. *Neurocomputing*, 51:161–180, 2003.

[8] Kunihiko Fukushima and Sei Miyake. Neocognitron: A new algorithm for pattern recognition tolerant of deformations and shifts in position. *Pattern Recognition*, 15(6):455–469, 1982.

[9] G. E. Hinton. Training products of experts by minimizing contrastive divergence. *Neural Computation*, 14(8):1771–1800, 2002.

[10] G. E. Hinton. A practical guide to training restricted boltzmann machines. *Momentum*, 9(1):599–619, 2010.

[11] G. E. Hinton, S. Osindero, and Y. W. Teh. A fast learning algorithm for deep belief nets. *Neural Computation*, 18:1527–1554, 2006.

[12] G. E. Hinton and R. R. Salakhutdinov. Reducing the dimensionality of data with neural networks. *Science*, 313:504–507, 2006.

[13] J. J. Hopfield. Neural networks and physical systems with emergent collective computational abilities. *Proceedings of the National Academy of Sciences of the USA*, 79(8):2554–2558, 1982.

[14] F. J. Huang and Y. LeCun. Large-scale learning with svm and convolutional nets for generic object categorization. *Proceedings of the 2006 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 1:284–291, 2006.

[15] D. H. Hubel and T. N. Wiesel. Receptive fields, binocular interaction and functional architecture in a cat's visual cortex. *Journal of Physiology (London)*, 160:106–154, 1962.

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

[17] Y. LeCun, F.J. Huang, and L. Bottou. Learning methods for generic object recognition with invariance to pose and lighting. *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2:97–104, 2004.

[18] Aleix M. Martínez. Recognizing imprecisely localized, partially occluded, and expression variant faces from a single sample per class. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(6):748–763, 2002.

[19] V. Nair and G. E. Hinton. 3d object recognition with deep belief nets. *Advances in Neural Information Processing Systems (NIPS)*, pages 1339–1347, 2009.

[20] M. Ranzato, J. Susskind, V. Mnih, and G. Hinton. On deep generative models with applications to recognition. *2011 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2857–2864, 2011.

[21] M. A. Ranzato, F. J. Huang, Y. L. Boureau, and Y. LeCun. Unsupervised learning of invariant feature hierarchies with applications to object recognition. *2007 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1–8, 2007.

[22] P. Smolensky. Information processing in dynamical systems: Foundations of harmony theory. In David E. Rumelhart and James L. McLelland, editors, *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*, volume 1, chapter 6, pages 194–281. MIT Press, 1986.

[23] P. W. M. Tsang and P. C. Yuen. Recognition of partially occluded objects. *IEEE Transactions on Systems, Man and Cybernetics*, 23(1):228–236, 1993.

[24] John Winn and Jamie Shotton. The layout consistent random field for recognizing and segmenting partially occluded objects. *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, 1:37–44, 2006.

[25] Laurenz Wiskott and Christoph Von Der Malsburg. A neural system for the recognition of partially occluded objects in cluttered scenes: A pilot study. *International Journal of Pattern Recognition and Artificial Intelligence*, 7(4):935–948, 1993.