Emotion Detection using Deep Belief Networks

Kevin Terusaki and Vince Stigliani

May 9, 2014

Abstract

In this paper, we explore the exciting new field of deep learning. Recent discoveries have made it possible to efficiently train artificial neural networks with multiple layers of hidden units, allowing for more hierarchical, incrementally abstracted representations. We study the performance of a Deep Belief Network, a powerful deep learning architecture, on the task of classifying the emotion of face images. We find that the DBN architecture autonomously learns representationally useful and human-interpretable features of the input, and that this method is still effective when pretrained with unlabeled data. Finally, we propose future extensions to refine our understanding of the many design choices involved in building a deep learning model.

1 Introduction

The discovery of efficient training algorithms for artificial neural networks (ANNs) stands as one of the most influential discoveries in the relatively short history of artificial intelligence and has led to many significant advances across machine learning and robotics [5]. The general idea, loosely inspired by the computational properties of biological neurons, involves a layered, forward-connected network of nodes composed of an input and output layer and possibly one or more hidden layers in between. The nodes at the input layer correspond to the low level information the network is trying to learn, such as pixel values of an image in a classification task or a sensorimotor representation of the environment in a robot control task. The output layer represents the desired target of learning—the category of an image or proper motor output. The hidden units, as their name describes, are not actually observed components of the learning problem, and so are less easily interpreted, but they allow the network to re-represent the input before sending it to the output. Hidden layers provide representation at a level of abstraction in between input and output, and greatly increase the expressivity of ANNs.

Since the hidden units are not directly observed, the burden of defining the number of units and layers lies on the programmer. A general heuristic is that the breadth and depth of the network should increase with the complexity of the learning task, but optimizing the specific size parameters is often a process of trial and error. However, a single sufficiently large hidden layer is theoretically capable of learning any problem that can be learned by an arbitrarily deep network. Moreover, researchers have shown that traditional ANN learning algorithms like back-propagation are not effective for training deeper networks.

In particular, Hochreiter described the vanishing gradient problem, which analyzed how error gradients decay exponentially so lower-level layers receive little to no information on how to change weights once the network becomes reasonably good at a task [4]; in effect, the lower levels are dispensable in deep architectures trained with back-prop. Because of these properties, most ANN approaches have favored shallow networks.

However, in more recent years, other modeling methods have gained prominence in machine learning as they outperform ANNs on a variety of complex tasks. In particular, most state-of-the-art systems have used methods like support vector machines (SVMs) and Bayesian modeling for the so-called 'hard AI' problems like language processing and vision. These problems can be characterized by a particularly large increase in abstraction between input and output. Some have suggested that the task complexity requires more incremental stages of rerepresentation than is achievable by shallow neural networks.

The last ten years have spawned what is perhaps the second great revolution of ANNs-deep learning. Deep learning is the study of efficiently training deep, densely connected networks. The breakthrough came from the lab of back-propagation pioneer Geoffrey Hinton with the discovery that pretraining deep-architectures layer-by-layer with unsupervised models like Restricted Boltzmann Machines (RBMs) provide a good initialization of network parameters that can then be fine-tuned with supervised data and the backprop training algorithm [2]. In this way, it can be seen as an extension of, rather than alternative to, traditional ANN approaches. Typically, initial network parameter settings are chosen randomly or hard-coded to reflect hypothesized intermediate features based on the programmer's domain knowledge. These relatively naïve methods become more problematic as the network deepens, where it increasingly important but also extremely difficult to identify initial features a priori at each level that will be useful in learning the task. Deep learning initializes the network in a parameter space where the error gradient more fully propagates through the network in fine-tuning, thus addressing the vanishing gradient problem of deep architectures. Deep learning methods have led to renewed prominence of ANNs and state-of-the-art performance on numerous 'hard AI' benchmark tasks.

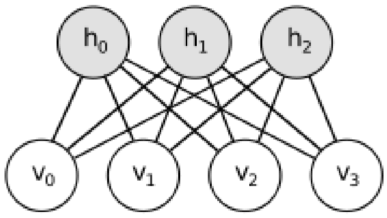

Figure 1: An illustration of the layers and connections of an RBM

A Deep Belief Networks (DBN) is one common way to implement a deep neural net. DBNs incrementally create the network topology layer-by-layer, starting with the input and first hidden layers. Each layer is trained with a Restricted Boltzmann Machine (RBM), which is a stochastic neural network composed of a visible layer and a hidden layer as illustrated in Figure 1. An RBM learns the probability distribution of a set of observations over the hidden units. Visible units correspond to the attributes of an observation (i.e. pixel values of an image), and hidden units model the dependencies between different aspects of the observed data, which can be interpreted as non-linear feature detection.

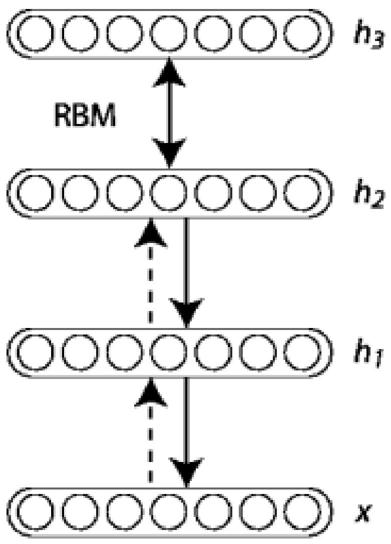

Figure 2: A simple graphical illustration of a DBN with 3 hidden layers

DBNs are formed by stacking a series RBMs, starting with the input layer and first hidden layer. Figure 2 shows an example of the DBN architecture. Once the first RBM is trained, its hidden layer becomes the visible layer of a second RBM and we add a new layer as the hidden layer of the new RBM. This entirely unsupervised pretraining can be continued indefinitely up to the final hidden layer. After this process is complete, connecting the final hidden layer to an output layer results in a deep neural network that can be trained with gradient descent on the error from the target output. In this way, a DBN iteratively learns increasingly abstract, hierarchical representations by composing the lower-order features of preceding layers.

In this paper, we explore the design and performance of DBNs through a series of experiments in facial emotion image classification. We believe that the levels of abstraction involved in mapping a matrix of pixel values to an emotion category make it particularly well suited for studying deep architectures. Moreover, given the compositional nature of faces (where individual features like eyes, lips, and forehead are combined in a more holistic structure), we expected that the learned intermediate representations would be more qualitatively interpretable.

We expect to see our Deep Belief Network develop low-level feature representations of the face with and combine these into higher-order representations to compose entire faces. Once these features have been learned in unsupervised training, we expect that the model will efficiently learn the particular classification task. Moreover, we anticipate that deeper architectures will outperform shallow networks, since our task was chosen so promote hierarchical representations. We have less refined predictions with regards to the influence of the size of hidden layers, but generally anticipate that forcing information compression by reducing the number of units in the hidden layer will result in better abstractions.

2 Experiment

We ran our experiments on the Karolinska Directed Emotional Faces (KDEF) dataset containing 4900 pictures of 70 individuals expressing seven unique human facial expressions. We only used the front-facing images narrowing down our dataset to 980 images. For our pre-training data, we use the KDEF dataset for one set of experiments. For another set of experiments we used the Faces In the Wild dataset for pre-training which contains over 13,000 images. Then, we used the KDEF dataset for fine-tuning the generative model and testing its correctness.

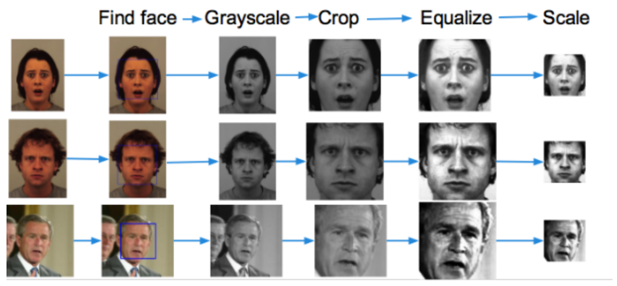

Figure 3: Visualization of our image preprocessing procedure. The top two rows are images from the labeled emotion dataset, and the bottom row is from the unlabeled Faces in the Wild dataset.

To simplify the learning problem, we developed the image preprocessing script illustrated in Figure 3. Using the openCV computer vision library, we extracted a face box from the background, grayscaled the image, equalized the histogram of grayscale values, and finally resized the image to 100×100 pixels. These 100×100 grayscaled face images were used as inputs to our system (grayscale values were scaled to be between 0 and 1). Our choice of image size was based on a rough estimate of the complexity of stimulus relative to the MNIST digit task, which uses an image size of 28×28 pixels.

We used Theano 0.6, a Python library that allows a user to efficiently define, optimize, and evaluate mathematical expressions involving multi-dimensional arrays. Theano incorporates GPU processing to speed up run-time. We used a Quadro FX 3800 GPU to run all experiments. In particular, we adapted code from Theano's Deep Learning Tutorial which contains a built-in Restricted Boltzmann Machines library to construct our Deep Belief Network.

We ran two sets of experiments: 1) Increasing the number of hidden nodes in each layer 2) Increasing the number of layers with a fixed number of hidden nodes. In both sets of experiments, we ran each experiment for 100 pre-training epochs and 400 fine-tuning epochs with a pre-training learning rate of 0.01, a fine-tuning learning rate of 0.1 and a batch size of 10. The 10,000 DBN inputs corresponds to the number of pixels in each 100 by 100 image. Therefore, each input represents a single pixel of the image. In experiment set (1), we ran experiments with the following number of hidden nodes–100_100, 200_200, 500_500 (each number represents the number of hidden nodes in that layer, therefore, each DBN contains two layers in experiment (1)). Additionally, we compared the results between runs that pre-trained on a larger, unlabeled dataset with those that pre-trained with a smaller, labeled dataset. In experiment set (2), we kept the hidden nodes constant at 500 and increased the hidden layers from two to three layers.

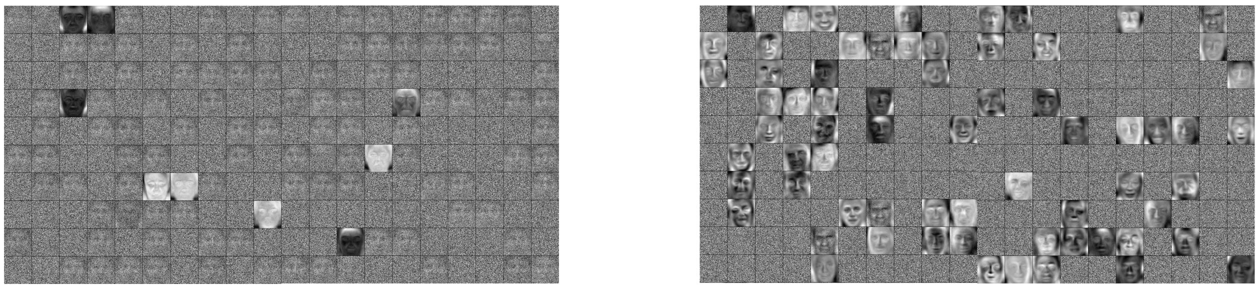
Figure 4: Best filters for 200_200 Deep Belief Network. Left image are filters pre-trained on smaller dataset. Right image are filters pre-trained on larger dataset.

3 Results

3.1 Qualitative Anaylsis

Figure 4 contains filters from our best 200_200 Deep Belief Network experiment. The left images are filters of pretraining on the smaller labeled dataset while the right images are filters of pretraining on the larger unlabeled dataset. These filter results suggest that pre-training requires many more images to start developing high-level feature recognition and fully make use of all the hidden nodes.

By examining a single filter, we can see the high-level feature detection that the DBN has developed through pre-training. Looking at Figure 5, we can see that this particular filter is focusing on the forehead region because of the high activations occurring on that part of the image. Higher activations correspond to whiter pixels while lower activations correspond to darker pixels. Figure 6 displays a filter that has high activations around the eye and chin area, suggesting that this filter is focusing on those areas. It is interesting to note that we hypothesized that the DBN would capture discrete features of faces (i.e.one feature of the face would have high activation while the rest of the face would be white noise). However, our results show that there is less noise in the filters than expected. In many of the filters there are distinguishable faces instead of noise.

3.2 Quantitative Analysis

We seperated our testing into two main types: pretraining with only the labeled images that are used in finetuning, and pretraining with the large corpus of

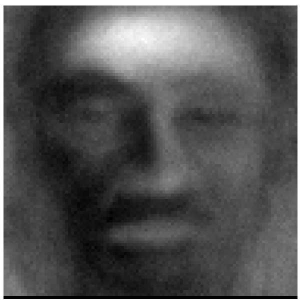
Figure 5: Filter focusing on forehead features

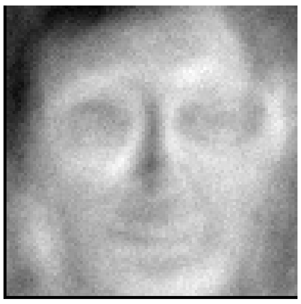
Figure 6: Filter focusing on eye features

unlabeled face images. For each of these pretraining sets, we considered the effects of increasing both hidden layer size and depth.

In our results, we report the validation error because in our dataset, this is equivalent to the test error since we randomly divided 20 percent of our total images into the validation set and 10 percent into the test set. Ideally, we would have liked to obtain test images from another dataset that uses an identical classification scheme. However, given our interest in exploring the DBN itself rather than the task, we believe that the trajectory of validation error over time provides a useful way to analyze DBN performance.

Figure 7 shows the results of changing the breadth of hidden layers. Due to a bug in the theano code, the DBN with 100 units per hidden layer did not learn at all. The network with 200 units ultimately performed at the level of the other two systems after about 300 epochs of fine-tuning. The 500 and 700 unit layers learned the fastest, and there was not an appreciable difference between the two. All three networks ultimately acheived around 30 percent accuracy after 400 epochs.

Since 500 units seemed sufficient to efficiently learn this task, we next explored topological variations on the 500 node hidden layer DBN. Figure 8 shows the results of this testing. We did not observe any significant differences between topologies in terms of final performance or rate of learning, although we do observe a slight lag in learning rate for the three hidden layer DBN in the early epochs. However, the fact that it very quickly achieves the same performance as the shallower networks suggests that the pretraining does indeed reduce the problem of the vanishing gradient for deeper networks. In future work, it would be interesting to note whether the deeper arhitecture produces more generalizable results and whether it is fully utilizing all the layers.

Next we considered the effect of pretraining with the much larger corpus of images from Faces in the Wild. Though the 13,000+ image set is ten times larger than our labeled pretraining set, it is still significantly less than the 60,000 images used to learn the seemingly less complex features of handwritten digits. Nevertheless, we expected to see benefits of including this unlabeled data in pretraining.

Figure 9 displays the results of unlabeled data pretraining on different 2-hidden-layer topologies. The DBN with 100 hidden units did not significantly improve from 60 percent error over 250 epochs. As the breadth of the hidden layer increases, we observe an increase in the early learning rate and a slight but

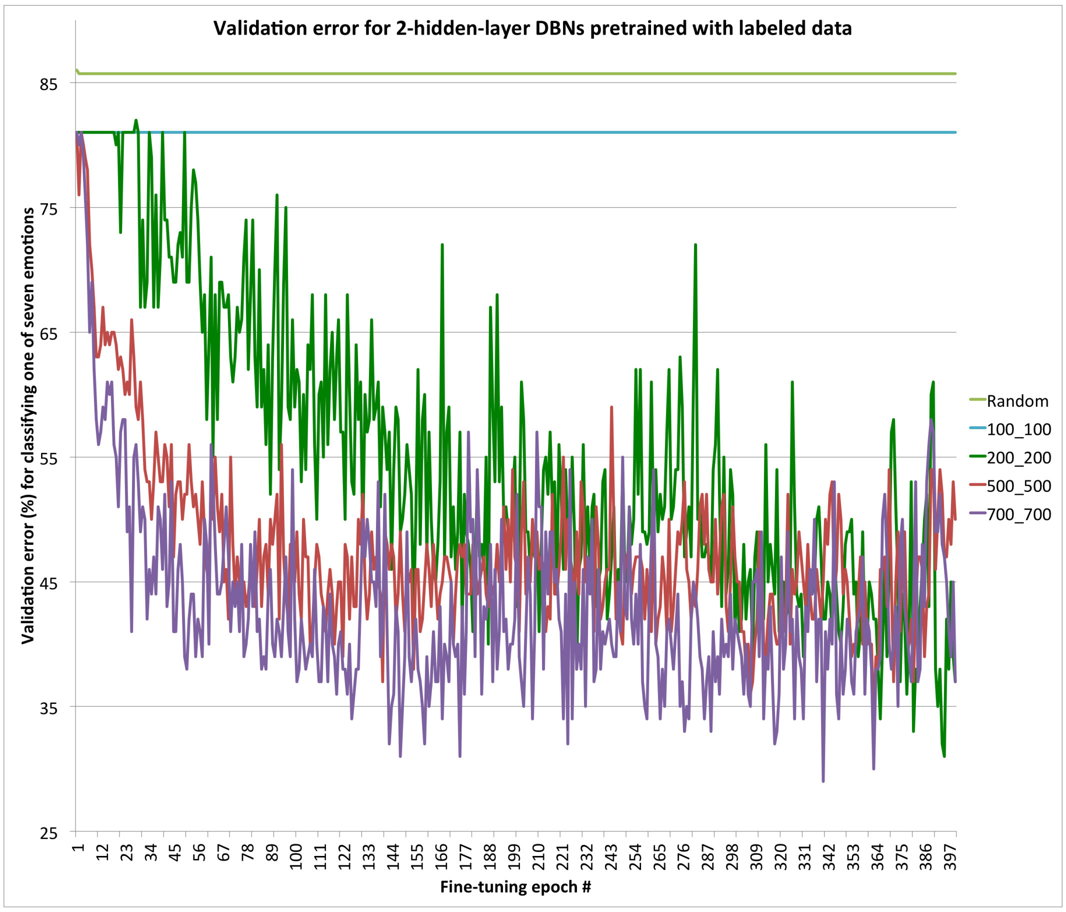

Figure 7: DBN validation error on emotion classification when pretrained with small labeled dataset. The label x_y indicates a DBN with x units in the first hidden layer and y units in the second layer

not significant difference in the final performance. Relative to DBNs pretrained with the smaller unlabeled dataset, we see a faster convergence of validation error, whereby the DBNs trained with unlabeled data reach an optimum in much fewer epochs.

Finally, we considered different variations on the 500-unit hidden layer topology. Results are shown in Figure 10. We do not see significant differences between the different configurations, although the learning progress seems substantially noisier in the DBN with 3 hidden layers. Even with our larger pretraining set, we likely do not have nearly enough pretraining data to fully realize the benefit of additional layers.

4 Conclusion

There are many design choices involved in training DBNs that we have not even begun to address. For example, we did not even begin to change the specific

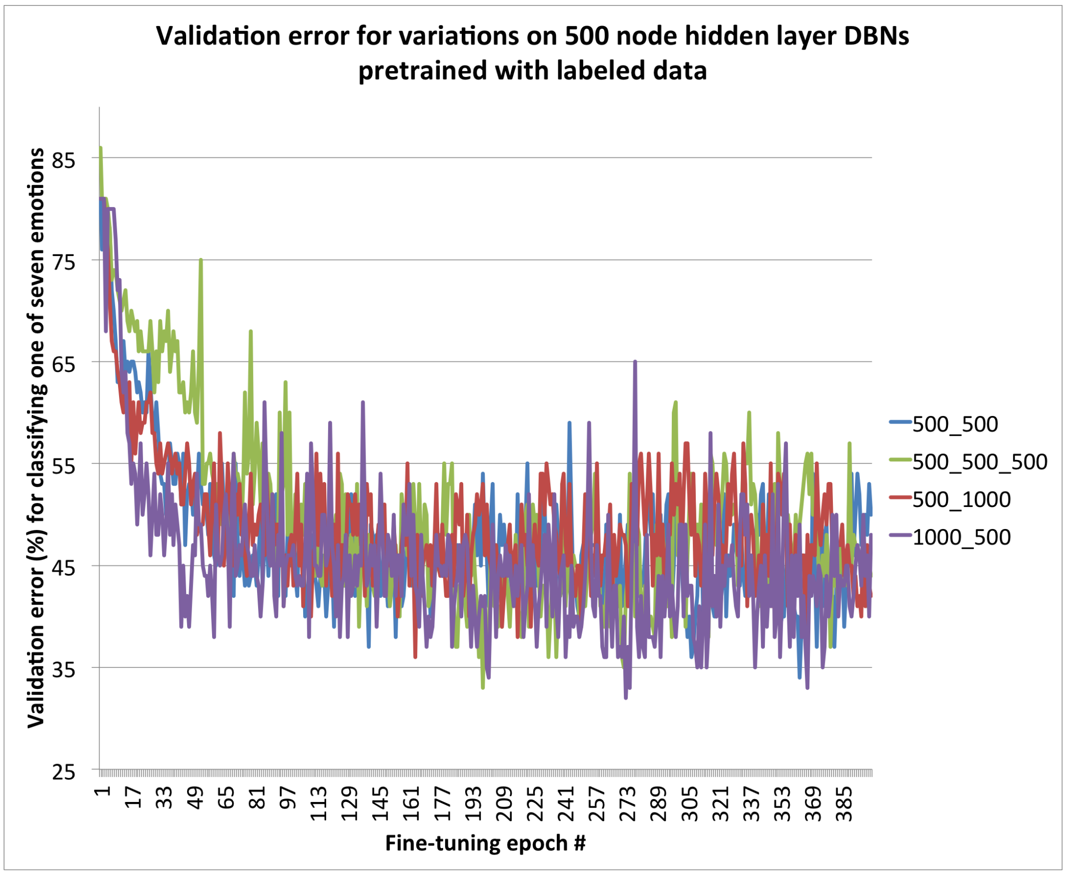

Figure 8: DBN validation error on emotion classification when pretrained with small labeled dataset with different variations of a 500 node hidden layer DBN

learning parameters. As a more theoretically interesting endeavor, it would be interesting to understand the importance of visual relatedness in pretraining and fine-tuning. Would it be useful to pretrain a face classifier with a broader corpus of images (such as a random sampling of images from a robot's camera as it interacts with the world) that are less semantically related to the final classification task. To this end, we imagine an embodied system with a generalized feature detector module of stacked RBMs trained with minimal to no supervision, combinded with individul top-level classifiers that can access these generalized hidden feature represenations for more fine-tuned classification as the need arises.

Since it is difficult to visualize the pre-training filters beyond the first layer, further research can be devoted to finding a good method to visualize the contribution of deeper hidden units. This would give users a better conceptualization of how the Deep Belief Network is aggregating features of a face to construct an emotion. Additionally, we would want to devote more time to a real-time emotion detection system that would use the trained Deep Belief Network to recognize an emotion through a webcam.

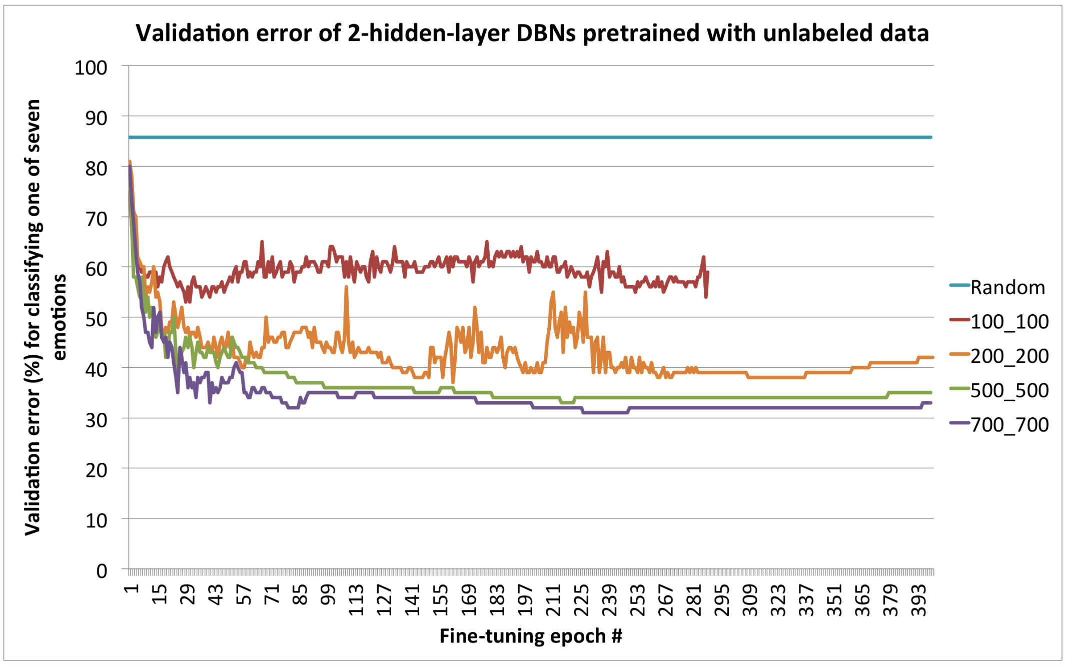

Figure 9: DBN validation error on emotion classification when pretrained with large unlabeled dataset with different hidden layer sizes

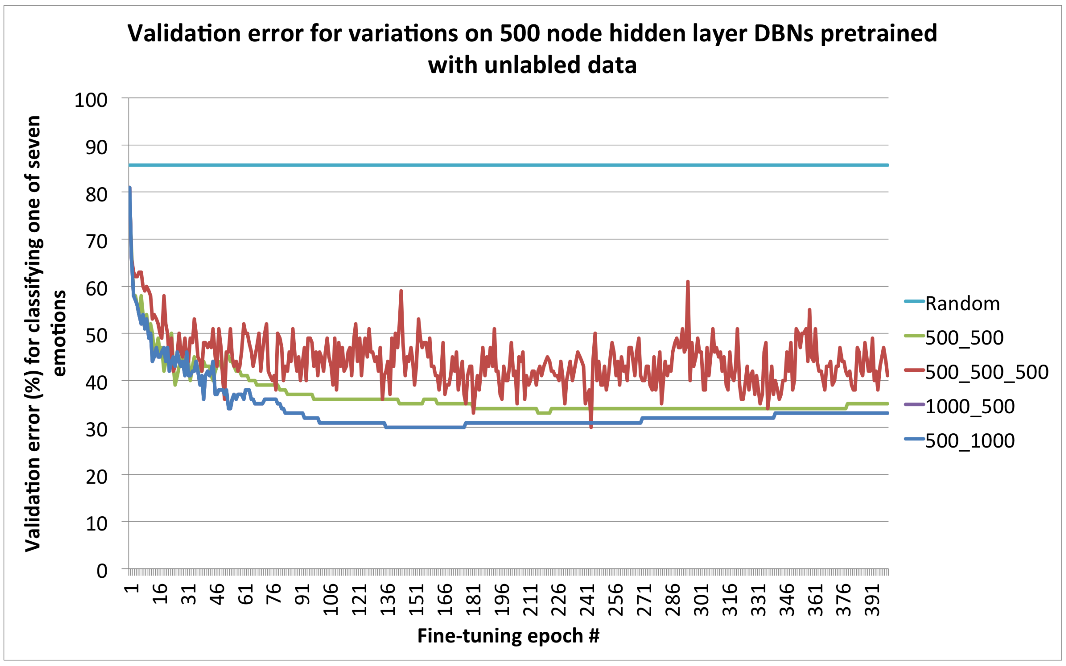

Figure 10: Validation error when pretrained with a large unlabeled dataset across different topological configurations

References

[1] J. Bergstra, O. Breuleux, F. Bastien, P. Lamblin, R. Pascanu, G. Desjardins, J. Turian, D. Warde-Farley and Y. Bengio. (2010). Theano: A CPU and GPU Math Expression Compiler. In *Theano: A CPU and GPU Math Expression Compiler*

[2] G. E. Hinton, Y. Teh. (2006). A fast learning algorithm for deep belief nets.

[3] G. E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, R. R. Salakhutdinov. (2012) Improving neural networks by preventing co-adaptation of feature detectors.

[4] S. Hochreiter. The Vanishing Gradient Problem During Learning Recurrent Neural Nets And Problem Solutions.

[5] T. Schultz. (1995). Chapters 1-2. In *Computational Developmental Psychology.*