Do Deep Nets Really Need to be Deep?

Lei Jimmy Ba
University of Toronto
jimmy@psi.utoronto.ca

Rich Caurana
Microsoft Research
rcaurana@microsoft.com

Abstract

Currently, deep neural networks are the state of the art on problems such as speech recognition and computer vision. In this extended abstract, we show that shallow feed-forward networks can learn the complex functions previously learned by deep nets and achieve accuracies previously only achievable with deep models. Moreover, the shallow neural nets can learn these deep functions using a total number of parameters similar to the original deep model. We evaluate our method on TIMIT phoneme recognition task and are able to train shallow fully-connected nets that perform similarly to complex, well-engineered, deep convolutional architectures. Our success in training shallow neural nets to mimic deeper models suggests that there probably exist better algorithms for training shallow feed-forward nets than those currently available.

1 Introduction

You are given a training set with 10M labeled points. When you train a shallow neural net with one fully-connected feedforward hidden layer on this data you obtain 85% accuracy on test data. When you train a deeper neural net as in [2] consisting of convolutional layers, pooling layers, and multiple fully-connected feedforward layers on the same data you obtain 90% accuracy on the test set.

What is the source of this magic? Is the 5% increase in accuracy of the deep net over the shallow net because: a) the deep net has more parameters than the shallow net; b) the deep net is deeper than the shallow net; c) nets without convolution can't learn what nets with convolution can learn; d) current learning algorithms and regularization procedures work better with deep architectures than with shallow architectures; e) all or some of the above; f) none of the above?

In this extended abstract we provide empirical evidence that shallow nets are capable of learning the same function as deep nets, and are able to do this with a total number of parameters comparable to the deep net. We do this by first training a state-of-the-art deep model, and then training a shallow model to mimic the deep model. The mimic model is trained using the model compression scheme described in the next section. Remarkably, with model compression we are able to train shallow nets to be as accurate as the deep models they are trained to mimic, even though we are not able to train shallow nets to be as accurate as the deep nets when the shallow nets are trained directly on the original labeled training data.

If a shallow net with the same number of parameters as the deep net can learn to mimic the deep net with high fidelity, then it is clear that the function learned by the deep net does not really have to be deep. This suggests that current training algorithms work better with deep architectures, and that with improved training procedures we might be able to train shallow models as accurate as deep models currently are.

2 Model compression

The main idea behind model compression [3] is to use train a compact model to approximate the function learned by a more complex model. For example, it has been demonstrated that a single neural net of moderate size often can be trained to mimic a much larger complex ensemble containing thousands to tens of thousands of models. The small neural net may contain 1000 times fewer parameters and execute a 1000 times faster, but often is just as accurate as the complex ensemble it is trained to mimic.

Model compression works by passing unlabeled data through the large, accurate model to collect the scores predicted by that model. These scores capture the function learned by the complex model at the sampled points. This synthetically labeled data set is then used to train the compression mimic model. Note that the mimic model is not trained on the original labels. Instead, it is trained to learn the function learned by the large, presumably more accurate model. If the mimic model could be trained perfectly it would make exactly the same mistakes as the complex model that was trained on the original data.

Surprisingly, usually it is not (yet) possible to train a small neural net on the original training data to be as accurate as the complex model. Model compression demonstrates that a small neural net could, in principle, learn the more accurate function, but with the current learning algorithms we are unable to train a model with that accuracy without training the complex accurate ensemble first and then training the neural net to mimic it. Clearly, when it is possible to mimic the function learned by a complex model with a much smaller net, the complex model wasn't truly that complex. This suggests that the complexity of a learned model, and the size of the representation best used to learn that model, are two different things.

3 TIMIT dataset

The TIMIT speech corpus has 462 speakers in the training set. There is a separate development set including 50 speakers for cross-validation, and a final test set with 24 speakers. The raw waveform audio data were pre-processed using 25ms Hamming window shifting by 10ms to extract Fourier-transform-based filter-banks with 40 coefficients (plus energy) distributed on a mel-scale, together with their first and second temporal derivatives. We included +/- 7 nearby frames to formulate the final 1845 dimension input vector. The data input features were normalised by subtracting the mean and dividing by the standard deviation on each dimension. All 61 phoneme labels represented in tri-state, 3 states for each one of the 61 phoneme and 183 dimensions target label vector, are were used during the training and decoding, then mapped to 39 classes as in [6] for scoring.

4 Deep Learning on TIMIT

Deep learning is first successfully applied to speech recognition in [7]. We follow the same framework and train two deep models on TIMIT, DNN and CNN. DNN is a deep neural net consisting of three fully-connected feedforward hidden layers consisting of 2000 rectified linear units (ReLU) [8] per layer. CNN is a deep neural net consisting of a convolutional layer and max-pooling layer followed by three hidden layers containing 2000 ReLU units.[1] Dropout is applied to both models preventing over-fitting.

The accuracy of DNN and CNN on the final test set is shown in table 1. The error rate of the convolutional deep net is about 2.1% better than the deep net. The table also shows the accuracy of a shallow neural net with 8000 hidden units (SNN). Despite having a similar number of parameters as the DNN, the shallow model is 2% less accurate than the DNN, and 4.1% less accurate than the CNN.

5 Training an SNN to Mimic a Deeper Model

We train the compression mimic SNN model using the data labeled by an ensemble of deep nets. The deep models are trained in the usual way using softmax output and cross-entropy cost function. The shallow SNN, however, instead of being trained with cross-entropy on the 183 p values where

$p_k = e^{z_k} / \sum_j e^{z_j}$ produced by the deep model, is trained directly on the log probability values z before the softmax activation. Training on the log probability value makes learning easier for the shallow neural network because it puts more emphasis on confident predictions so the shallow net will try to get data points with high confidence predictions correct first during learning.

We formulate the learning objective function as a regression problem given training data $\{(x^{(1)}, z^{(1)})$,...,$(x^{(T)}, z^{(T)})\}$:

$$\mathcal{L}(W, \beta) = \frac{1}{2T} \sum_t ||g(x^{(t)}; W, \beta) - z^{(t)}||_2^2, \tag{1}$$

where, W is the weight matrix between input features x and hidden layer, β is the weights from hidden to output units, $g(x^{(t)}; W, \beta) = \beta f(Wx^{(t)})$ is the model prediction on the t^{th} training data point and $f(\cdot)$ is the non-linear activation of the hidden units. The parameters W and β in shallow neural networks are updated using error back-propagation algorithm and stochastic gradient descent with momentum.

6 Speed up learning by including a linear layer

To match the similar number of parameters as in deep nets, shallow nets have to put all the non-linear hidden units in a single layer that results in a large weight matrix W. When training a large shallow neural network with tens of thousands hidden units, we find it is very slow to learn the large number of parameters in the weight matrix between input and hidden layer of the size $O(HD)$, where D is input feature dimension and H is the number of hidden units. It is challenging to estimate this large weight matrix using gradient descent methods due many highly correlated parameters so the learning algorithm converges slowly. We also notice that during learning, shallow nets spends most of the computation in the costly matrix multiplication of the input data vectors and large weight matrix.

To limit the information flow through the network and the number of effective parameters, we introduce a bottleneck linear layers with k hidden units between the input and the non-linear hidden layer. Alternatively, we can factorize the weight matrix $W \in \mathbb{R}^{H \times D}$ into the product of two low rank matrices, $U \in \mathbb{R}^{H \times k}$ and $V \in \mathbb{R}^{k \times D}$, where $k << D, H$. The new cost function can be written as:

$$\mathcal{L}(U, V, \beta) = \frac{1}{2T} \sum_t ||\beta f(UVx^{(t)}) - z^{(t)}||_2^2 \tag{2}$$

The weights U and V can be learnt by back-propagating through the linear layer. This re-parameterization of weight matrix W not only increases the convergence rate of the mimic shallow neural networks but also reduces the memory space from $O(HD)$ to $O(k(H + D))$. The reduced memory usage enables us to train large shallow models that were previously unfeasible due to excessive memory usage. Because the linear layer can always be absorbed into the weight matrix W, even after adding the linear layer, the model still keeps the same representational power as a normal shallow net. The bottleneck linear layer is essential to successfully train large shallow nets in our experimental results.

7 Experimental results

We trained shallow neural nets and deep neural nets using cross-entropy cost function on the original 0/1 labels. We show a comparison, in table 1, of these models with the mimic shallow neural networks trained using model compression technique on phoneme recognition performance task. We used the same architecture for the convolutional neural net as in [4]. The exact DNN and CNN models in the table are used to label the training data for the mimic SNNs. The log probability of the prediction values of the training input features from the deep models are used as regression target to train mimic SNN. We also trained a large SNN with 400k non-linear hidden units combining a linear layer with 250 linear units. This large model is trained on the target produced by an ensemble of different CNN models. During our evaluation, the insertion penalty and the language model weighting in the decoder are always fixed to 0 and 1 respectively. We did not tune these two hyper-parameters for our experiments and further improvement in performance can be achieved by picking those hyper-parameter values using the development set.

	architecture	# parameters	PER
DNN	$3 \times$ 2k hidden units with dropout	$\sim$12Million	21.6%
CNN	conv. and pooling layer followed by $3 \times$ 2k hidden units with dropout	$\sim$13Million	**19.5%**
SNN	1845-8000-183 with dropout	$\sim$12Million	23.6%
SNN mimic DNN	1845-8000-183	$\sim$12Million	21.7%
SNN mimic CNN	1845-8000-183	$\sim$12Million	20.6%
SNN mimic an ensemble model	1845-400k-183	$\sim$180Million*	**19.5%**

Figure 1: Comparison of shallow and deep models: phone error rate on TIMIT core test set

7.1 Discussion

We notice that in the table 1 the mimic shallow nets are able to perform just as well as their deep counter-parts when learnt using model compression technique. Normally, deep neural nets are much more competitive than the shallow nets when training with the current learning algorithms and cross-entropy cost function. We see that by increasing number of hidden layers from one to three, the phoneme recognition performance jumped by 2%, from 23.6% to 21.6%. Moreover, a five-layer CNN with highly engineered first and second layer further improve the performance to 19.5%. Even with dropout regularization [5] and smart weight initialization [9], the shallow nets trained on the original data are not able to come close to the performance of the deep models and suffer from over-fitting when increasing model size. When we train the SNNs using targets produced by other higher accurate models, namely DNN and CNN, the mimic SNNs are able to match the performance and perform just as well as a deep fully connected neural net with comparable number of parameters. Also, if we increase the model complexity, our fully connected SNN with 400k hidden units that has no built-in topology is able to reach the performance of a deep convolutional neural net that was complex and high engineered with prior structure and weight sharing property. It shows that we need to increase the number of the parameters in shallow models exponentially in order to capture complicated functions, e.g. deep convolutional nets.

Interestingly, the performance of the shallow nets increases along with the performance of the model it tries to approximate. In other words, if we increase the performance of the model that is used to generate training labels for SNN, the performance of the exact same mimic SNN with the same number of parameters will also increase accordingly.

8 Conclusions

We find that when training on log probability using compression technique and applying a particular constrains on model parameters, shallow neural networks can be learnt to achieve the performance that is previously achievable only by deep models. We evaluate our method on phoneme recognition tasks using TIMIT dataset. The shallow fully connected network trained using our method performs similarly comparing to well-engineered complex deep convolutional architectures. Finally, our strong results on TIMIT suggest new learning algorithms to train feed-forward networks to achieve better performance.

References

[1] Ossama Abdel-Hamid, Li Deng, and Dong Yu. Exploring convolutional neural network structures and optimization techniques for speech recognition.

[2] Ossama Abdel-Hamid, Abdel-rahman Mohamed, Hui Jiang, and Gerald Penn. Applying convolutional neural networks concepts to hybrid nn-hmm model for speech recognition. In *Acoustics,*

Speech and Signal Processing (ICASSP), 2012 IEEE International Conference on, pages 4277–4280. IEEE, 2012.

[3] Cristian Bucilu, Rich Caruana, and Alexandru Niculescu-Mizil. Model compression. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 535–541. ACM, 2006.

[4] Li Deng, Jinyu Li, Jui-Ting Huang, Kaisheng Yao, Dong Yu, Frank Seide, Michael Seltzer, Geoff Zweig, Xiaodong He, Jason Williams, et al. Recent advances in deep learning for speech research at microsoft. *ICASSP 2013*, 2013.

[5] G.E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, and R.R. Salakhutdinov. Improving neural networks by preventing co-adaptation of feature detectors. *arXiv preprint arXiv:1207.0580*, 2012.

[6] K-F Lee and H-W Hon. Speaker-independent phone recognition using hidden markov models. *Acoustics, Speech and Signal Processing, IEEE Transactions on*, 37(11):1641–1648, 1989.

[7] Abdel-rahman Mohamed, George E Dahl, and Geoffrey Hinton. Acoustic modeling using deep belief networks. *Audio, Speech, and Language Processing, IEEE Transactions on*, 20(1):14–22, 2012.

[8] V. Nair and G.E. Hinton. Rectified linear units improve restricted boltzmann machines. In *Proc. 27th International Conference on Machine Learning*, pages 807–814. Omnipress Madison, WI, 2010.

[9] Ilya Sutskever, James Martens, George Dahl, and Geoffrey Hinton. On the importance of initialization and momentum in deep learning.