Deep vs. Wide: Depth on a Budget for Robust Speech Recognition

Oriol Vinyals¹², Nelson Morgan¹²

¹International Computer Science Institute, Berkeley, California
²Department of Electrical Engineering and Computer Science, University of California at Berkeley

vinyals@icsi.berkeley.edu, morgan@icsi.berkeley.edu

Abstract

It has now been established that incorporating neural networks can be useful for speech recognition, and that machine learning methods can make it practical to incorporate a larger number of hidden layers in a “deep” structure. Here we incorporate the constraint of freezing the number of parameters for a given task, which in many applications corresponds to practical limitations on storage or computation. Given this constraint, we vary the size of each hidden layer as we change the number of layers so as to keep the total number of parameters constant. In this way we have determined, for a common task of noisy speech recognition (Aurora2), that a large number of layers is not always optimum; for each noise level there is an optimum number of layers. We also use state-of-the-art optimization algorithms to further understand the effect of initialization and convergence properties of such networks, and to have an efficient implementation that allows us to run more experiments with a standard desktop machine with a single GPU.

Index Terms: deep learning, neural networks, robust speech recognition

1. Introduction

In the past decade, the term “deep learning” has been popularized thanks to its various successes in many applications in machine learning such as speech or vision. In the speech community, deep learning has achieved enough maturity so that a number of commercial applications use it as part of their acoustic or language models (some completely replacing mixture models).

Much of this progress has built on technology that was developed years ago. In particular, the hybrid HMM/MLP system, in which the output posteriors have been used (after division by class priors) as scaled likelihoods for HMMs, was developed over twenty years ago [1, 2, 3]. Alternatively, the tandem approach, in which the MLP outputs are used (typically after a few transformations) as features for an HMM/GMM system, has been used for many tasks. Still, the newer systems have advanced beyond these earlier ones, primarily in 3 ways: (1) while most (but not all [4, 5, 6]) of the earlier systems used monophone classes, the newer ones tend to use finer categories; (2) the most obvious difference is that the newer approaches use many hidden layers instead of one or two, which is partly made possible by (3) machine learning approaches to initialize the networks so that the layers closer to the input still perform some useful task despite being so “fat” from the correction signals at the output.

Here we focus on the 2nd and 3rd aspects, since it has been taken as “common wisdom” that adding a significant number of layers is necessary for good performance. A number of papers have compared a “deep” network implementation to a standard MLP with a single large hidden layer, essentially always showing significant improvements from adding depth. However, we have not observed any careful study of the error rates for ASR when the number of parameters is kept constant over the different sized nets; in other words, determining the tradeoff between the depth and width of the network.

For the purpose of this paper, we are using the Aurora2 task [7], which uses spoken digit strings with natural noises artificially added to achieve different SNRs. There are many limitations to this data set: it is small, it does not incorporate properties of natural conversational speech, adding noise artificially does not provide natural human signal modifications in the presence of noise (e.g., Lombard effect), and additionally the test set has been significantly plumbed, having been used by researchers for a decade, so the raw error rates are not particularly relevant. All this being said, it is a commonly used data set, and once the reader sees that our performance is in a reasonable range, he or she can focus on the comparative results – how performance varies for the corresponding depths and widths.

2. Related Work

Since the related work in this area is pretty vast, we further divide this section in subsections.

2.1. Deep Learning before Deep Learning

As noted above, neural networks have been used for speech recognition for some time; in fact, there were early experiments at Stanford by B. Widrow and his students in the 1960s [8]. In the 1980s, a number of laboratories in Europe, the U.S., and Japan revived work on using neural networks to classify speech categories, and ultimately to generate speech category probabilities for use with hidden Markov models, which by then were the dominant method for dealing with the sequence of sounds in speech. In general these networks used a single hidden layer and a modest number of categories, although as noted above there were notable exceptions. For example, the HNN/ACID approach of J. Fritsch [4] used a tree of networks in order to estimate a large number of context-dependent classes, using the simple factoring trick expounded in [9]. Input to the Fritsch system was processed by a number of networks in order to derive probability estimates at the leaves, and thus was an example of an extremely deep network that was also context-dependent. It performed quite well on a number of large vocabulary tasks at the time. This network and many others were examples of the hybrid HMM/MLP approach. However, by 2000 it had become quite difficult for such systems to keep up competitively with the plethora of engineering advances that were developed.
for HMM/GMM systems, given the huge number of excellent laboratories that focused on improving the latter. The turning point for some of us was the Aurora task, in which we were required to use an HMM/GMM system with fixed characteristics and only could modify the front end. And so the Tandem approach [10] was adopted by many of us, in which the same networks we had developed were now used to generate features for an HMM/GMM system. This permitted researchers to continue to develop neural network approaches while taking advantage of advances in HMM/GMM systems.

Using these approaches, other systems were developed in which networks trained for an intermediate goal were then incorporated in larger networks, resulting in a deep structure; for example, a hierarchical system focused on temporal modulations of the spectrum [11] and systems that trained networks to focus on temporal characteristics within each critical band and then combined these networks with another network, e.g., [12]. Thus, many experiments were done in which some input variables (e.g., MFCC or mel spectra) were processed by multiple layers of artificial neurons prior to use by an HMM of some form. However, all of these techniques were largely relegated to providing an assist to a larger HMM/GMM system prior to the recent revival of hybrid HMM/MLP systems that has gone under the name of “deep neural networks” of various kinds.

2.2. Renaissance of Neural Networks in ASR

One of the problems with MLPs is that the objective function is non-convex, and as more hidden layers are added, finding a good local minimum becomes more challenging. Motivated by this problem, Deep Belief Networks were introduced in [13] and have been applied in several fields such as computer vision (see [13]), phone classification (see [14, 15]), and speech coding (see [16]). The new idea is to train each layer independently in a greedy fashion, by sequentially using the hidden variables as observed variables to train each layer of the deep structure. Initial success using DBNs on fairly small datasets and using small models quickly evolved, and the positive results of having deep models that yielded state-of-the-art acoustic modeling was adopted and generalized in many tasks and with much larger models [17, 18, 19, 20]. In the next section, we discuss previous work that focuses on the analysis and understanding of why neural networks have seen such a renaissance for acoustic modeling.

2.3. Prior Analysis Efforts

Since our paper attempts to explain, in a limited context, the main contributing factors that make deep learning successful are, we next review some of the previous work that discussed such key points in chronological order.

In [17], many interesting points were raised. First, in a large system such as the one that was used there (with 2000 hours), having triphone units as targets instead of monophones (which were more commonly use in previous work, with exceptions, as noted above) may have been the main contributing factor to the results obtained (4% absolute gains). Adding layers generally resulted in better accuracy, but the number of parameters was increased with every layer added, so that it was not entirely clear what was the main contributing factor to the good results - the depth, or the larger number of parameters. However, a flat (or shallow) model having the same parameters as the deepest model was also trained, which was 2% worse than the deep model. Additionally, pretraining added 2% in accuracy.

In [18], another method for pretraining was used, and again the depth of the model was demonstrated to be one of the key contributions to the 30% relative improvement. An important contribution in that work was the effect of pretraining. For smaller tasks such as TIMIT or MNIST, pretraining was found important as, besides helping the optimization of a deep architecture, provided a form of regularization to the network by using unsupervised learning. In the work by Seide et al, the architecture rather than pretraining seemed to be more important, and other forms of pretraining such as discriminative pretraining was able to perform as well as RBM pretraining.

Recent work in [21, 22] analyzed the input to the networks, and showed that log filter banks may be a better input instead of other transformations (e.g., MFCC), which have been the mainstream feature in classical GMMs due to some assumptions such as diagonal covariance.

3. Factors in Deep Learning

Since many researchers have been studying deep learning, we want to further discuss key points that, although widely discussed in the community, are important to both the speech community, and other applied machine learning fields.

3.1. Depth

The main point of this work is to analyze how the depth of the model affects the performance in Aurora2 recognition. To this effect, we consider a standard neural network with sigmoid hidden units and an output layer consisting of softmax that maps real numbers to probabilities. As discussed in Section 2, most of the recent success of neural networks has been attributed to having deep architectures (i.e. having many hidden layers, e.g., 5 or more). In [15, 18, 17] the effect of adding more layers is shown to improve performance by up to a few percent (absolute), but often adding more layers adds more parameters to the network, thus allowing for a more powerful model. In this paper, we do a fair comparison varying the number of hidden layers while preserving the complexity of the model (in terms of the number of parameters). As noted in the abstract, a limitation on the number of parameters can often be a proxy for realistic constraints on storage or computation, or in some cases for cost.

3.2. Initialization

Although pretraining was originally proposed as a better starting point to the optimization to find the “optimal” parameters to the network, later work suggested that other simpler initialization schemes were quite successful by studying the effect of vanishing gradients [23]. However, since pretraining in a small dataset such as Aurora2 still may have its benefits, all the results reported here use standard RBM initialization. Due to lack of space, we do not report results without pretraining, but even given the size of the dataset it did not seem to have a large effect — overall, the results for deeper models were the same except for models deeper than 4 layers, where on average they were 2% absolute worse).

3.3. Optimization

Instead of using standard gradient descent methods, we used an improved algorithm based on the idea behind Hessian Free optimization [24, 25]. These methods have already proven successful within the speech community (e.g. [26, 25]), and differ from other common methods in two ways:
Table 1: Table of Word Error Rates (WER) for Aurora2 experiments. The MFCC column corresponds to the use of MFCCs plus first and second derivatives as the features for the HMM. The others correspond to network features with each column corresponding to the indicated number of hidden layers. In each case the resulting features are appended to the MFCC-based features. The AVG row gives the average of the WERs for all the SNRs, while the UsableAVG row is the average of WERs for all the SNRs of 10 dB and higher.

<table>
<thead>
<tr>
<th></th>
<th>MFCC</th>
<th>1-hid</th>
<th>2-hid</th>
<th>3-hid</th>
<th>4-hid</th>
<th>5-hid</th>
<th>6-hid</th>
<th>7-hid</th>
<th>8-hid</th>
</tr>
</thead>
<tbody>
<tr>
<td>clean</td>
<td>0.97%</td>
<td>0.61%</td>
<td>0.51%</td>
<td>0.53%</td>
<td>0.55%</td>
<td>0.55%</td>
<td>0.46%</td>
<td>0.52%</td>
<td>0.51%</td>
</tr>
<tr>
<td>20dB</td>
<td>4.84%</td>
<td>2.27%</td>
<td>1.80%</td>
<td>1.89%</td>
<td>2.16%</td>
<td>1.98%</td>
<td>2.24%</td>
<td>2.00%</td>
<td>2.09%</td>
</tr>
<tr>
<td>15dB</td>
<td>14.65%</td>
<td>4.67%</td>
<td>4.25%</td>
<td>4.34%</td>
<td>4.83%</td>
<td>4.60%</td>
<td>5.11%</td>
<td>4.64%</td>
<td>5.00%</td>
</tr>
<tr>
<td>10dB</td>
<td>35.80%</td>
<td>12.05%</td>
<td>10.53%</td>
<td>11.08%</td>
<td>11.75%</td>
<td>10.92%</td>
<td>11.81%</td>
<td>10.87%</td>
<td>11.81%</td>
</tr>
<tr>
<td>5dB</td>
<td>65.78%</td>
<td>29.20%</td>
<td>26.23%</td>
<td>26.57%</td>
<td>27.47%</td>
<td>25.94%</td>
<td>27.02%</td>
<td>24.61%</td>
<td>26.45%</td>
</tr>
<tr>
<td>0dB</td>
<td>86.33%</td>
<td>58.35%</td>
<td>54.42%</td>
<td>53.02%</td>
<td>53.85%</td>
<td>52.67%</td>
<td>53.09%</td>
<td>49.67%</td>
<td>52.01%</td>
</tr>
<tr>
<td>neg5dB</td>
<td>92.75%</td>
<td>83.94%</td>
<td>80.32%</td>
<td>80.12%</td>
<td>81.26%</td>
<td>79.72%</td>
<td>79.09%</td>
<td>76.06%</td>
<td>78.61%</td>
</tr>
<tr>
<td>AVG</td>
<td>43.02%</td>
<td>27.30%</td>
<td>25.44%</td>
<td>25.37%</td>
<td>25.98%</td>
<td>25.20%</td>
<td>25.55%</td>
<td>24.06%</td>
<td>25.21%</td>
</tr>
<tr>
<td>UsableAVG</td>
<td>14.06%</td>
<td>4.90%</td>
<td>4.27%</td>
<td>4.46%</td>
<td>4.83%</td>
<td>4.51%</td>
<td>4.91%</td>
<td>4.51%</td>
<td>4.85%</td>
</tr>
</tbody>
</table>

- The second order information deals (in part) with the vanishing gradients problem. One of the effects of this, besides achieving faster convergence rates, is to greatly reduce the importance of pretraining.
- The fact that we use a step size tuned on a separate random batch of data than the one used to compute the gradient has good implications for reducing generalization error. As a result, we do not require early stopping as we do not observe overfitting, even with more parameters than samples. We still confirm this by observing errors on an independent cross-validation set.

Training was performed on a single GPU (NVIDIA GTX 580), and was run for 50 epochs (i.e. 50 passes through all the data), taking about 2 hours to complete. No overfitting was observed (i.e., cross validation likelihood kept improving every epoch), even for a network of 2.4M parameters trained on 1.4M samples.

4. Experimental Setup

For this paper, we use the Aurora 2 data set described in [7], a connected digit corpus which contains 8,440 sentences of clean training data and 56,056 sentences of clean and noisy test data. The test set comprises 8 different noises (subway, babble, car, exhibition, restaurant, street, airport, and train-station) at 7 different noise levels (clean, 20dB, 15dB, 10dB, 5dB, 0dB, -5dB), totaling 56 different test scenarios, each containing 1,001 sentences. Since we are interested in the performance of various deep learning architectures in mismatched conditions, all systems were trained only on the clean training set but tested on the entire test set. The inputs to each network consist of a context window of 9 frames of PLP-12 (13 parameters including C0) and its first and second derivatives; that is, 9 x 39 input variables. The hidden layers are all the same size, and are adjusted in length to make the number of parameters the same for each experiment. The outputs of each network consist of 55 units corresponding to phonetic categories for training. During HMM training and recognition, these outputs are used in Tandem mode, that is, used as features after taking the logarithm, transforming with PCA, and appending to an additional 39 features comprising MFCC-12 (including C0) and its first and second derivatives.

The parameters for the HTK decoder used for this experiment are the same as that for the standard Aurora2 setup described in [7]. The setup uses whole word HMMs with 16 states with a 3-Gaussian mixture with diagonal covariances per state; skips over states are not permitted in this model. This is the setup used in the ETSI standards competition.

The basic architecture considered for the DNN was inspired by the MNIST hand written digit recognition task [13] and the baseline is the same that we used in recent work (further details on the training procedure can be found in [27]).

5. Results

Figure 1 shows the frame accuracy (without an HMM) on a held out set of 800 utterances for cross validation. To our surprise, the shallow network seems to perform better than any of the deep variants (even when pretraining is used) when fixing the number of parameters to 2.4M (the same as in [27]). However, it is important to note that the cross validation data comprises clean speech only, and that the only source of variation is speaker variability. Furthermore, pretraining has a bigger impact with networks of depth 3 and more, as observed by the fact that without it, networks tend to perform worse as we go deeper. Although not plotted, the training curves show that training error actually improves with depth (reaching a minimum at 4 hid-
6. Conclusions

There are many follow-up experiments that should be run in order to better understand our results. In keeping with modern practice in “deep” network usage, we have used many more parameters than the more standard rules of thumb would suggest – more than one parameter per data pattern. Our view was that this gave the “deep” approaches their best chance to do well. However, the design space should be further explored, for instance by using many more parameters, as well as by using many fewer (we did also experiment with half and double the amount of parameters, but the conclusions with those models would have been the same). In the results reported here, we always used RBM retraining, which has been shown to be effective for such tasks; however, we have also been running experiments without the pretraining, and discriminant methods can provide alternative initialization of the layers. For simplicity’s sake, we used the network outputs in Tandem mode, and it would be good to try the same test using the hybrid HMM/MLP approach. The network architecture that we focus on here uses the same number of units per hidden layer, although we have other experiments running with different shapes. Finally, the task is relatively small, and we do not know if there might be different results for speech that is more variable (e.g., conversational), as well as with a much larger amount of training data.

Despite these intriguing possibilities for future work, what has already been done certainly does suggest that, given a particular parameter budget, it is not at all obvious that resources need to be allocated to a very large number of layers. It is reasonable to conclude that researchers and developers who are working on a new task should compare results for different depths using the maximum number of parameters that they deem practical for their application (assuming that they are using some reasonable method to check for overfitting) and, keeping that fixed, do comparisons such as those reported here. They may be quite surprised to find that a “middling” depth is quite sufficient.

7. Acknowledgements

We would like to acknowledge the Microsoft Research Fellowship for supporting Oriol Vinyals, and Suman Ravuri for useful discussions on the topic of the paper.

8. References