Available online at www.sciencedirect.com

ScienceDirect

Computer Speech and Language 41 (2017) 195–213

COMPUTER SPEECH AND LANGUAGE

www.elsevier.com/locate/csl

Building DNN acoustic models for large vocabulary speech recognition

Andrew L. Maas *, Peng Qi, Ziang Xie, Awni Y. Hannun, Christopher T. Lengerich, Daniel Jurafsky, Andrew Y. Ng

Stanford University, Stanford, CA 94305, USA

Received 23 January 2015; received in revised form 16 November 2015; accepted 23 June 2016
Available online 18 July 2016

Abstract

Understanding architectural choices for deep neural networks (DNNs) is crucial to improving state-of-the-art speech recognition systems. We investigate which aspects of DNN acoustic model design are most important for speech recognition system performance, focusing on feed-forward networks. We study the effects of parameters like model size (number of layers, total parameters), architecture (convolutional networks), and training details (loss function, regularization methods) on DNN classifier performance and speech recognizer word error rates. On the Switchboard benchmark corpus we compare standard DNNs to convolutional networks, and present the first experiments using locally-connected, untied neural networks for acoustic modeling. Using a much larger 2100-hour training corpus (combining Switchboard and Fisher) we examine the performance of very large DNN models – with up to ten times more parameters than those typically used in speech recognition systems. The results suggest that a relatively simple DNN architecture and optimization technique give strong performance, and we offer intuitions about architectural choices like network depth over breadth. Our findings extend previous works to help establish a set of best practices for building DNN hybrid speech recognition systems and constitute an important first step toward analyzing more complex recurrent, sequence-discriminative, and HMM-free architectures.

Keywords: Hidden Markov model deep neural network (HMM-DNN); Neural networks; Acoustic modeling; Speech recognition; Large vocabulary continuous speech recognition (LVCSR)

1. Introduction

Deep neural network (DNN) acoustic models have driven tremendous improvements in large vocabulary continuous speech recognition (LVCSR) in recent years. Understanding what design decisions lead to successful DNN-based speech recognizers is therefore a crucial analytic goal. For example initial research hypothesized that DNNs work well because of unsupervised pre-training (Dahl et al., 2012). However, DNNs with random initialization yield state-of-the-art LVCSR results for several speech recognition benchmarks (Hinton et al., 2012; Kingsbury et al., 2012; Vesel et al., 2013). Instead, it appears that modern DNN-based systems are quite similar to long-standing neural network acoustic modeling approaches (Bourlard and Morgan, 1993; Hermansky et al., 2000; Renals et al., 1994). Modern

* Corresponding author at: Stanford University, Stanford, CA 94305, USA.
Email address: amaas@cs.stanford.edu (A.L. Maas).

http://dx.doi.org/10.1016/j.csl.2016.06.007
0885-2308/

DNN systems build on these fundamental approaches but utilize increased computing power, training corpus size, and function optimization heuristics.

While the need for analysis is clear, and recent research on DNN acoustic models for LVCSR explores variations in network architecture, optimization techniques, and acoustic model training loss functions, system differences across research groups make it difficult to develop actionable conclusions. It is hard to determine whether, for example, a performance improvement is due to a better neural network architecture or a different optimization technique. Furthermore, DNN acoustic models in LVCSR are not simply classifiers, but are instead one sub-component of the larger speech transcription system. There is a complex relationship between downstream task performance, word error rate (WER), and the proximal task of training a DNN acoustic model as a classifier. Because of this complexity, it is unclear which improvements to DNN acoustic models will ultimately result in improved performance across a range of LVCSR tasks.

In this paper we offer a large empirical investigation of DNN performance on two LVCSR tasks in an attempt to establish a set of best practices for building DNN acoustic models. To further make careful comparison possible we restrict our focus to feed-forward DNN models trained with cross entropy, because they are the foundation of neural network acoustic models. A deep understanding of these feedforward DNNs can help guide the emerging work utilizing a wide variety of new architectures based on recurrent neural networks (Graves et al., 2013; Li and Wu, 2015; Sak et al., 2014; Vinyals et al., 2012; Weng et al., 2014), sequence-discriminative training (Kingsbury et al., 2012; Vesel et al., 2013; Wiesler et al., 2015), and HMM-free neural network approaches (Graves and Jaitly, 2014; Maas et al., 2015). Further, we seek to understand which aspects of DNN training have the most impact on downstream task performance. This knowledge can guide rapid development of DNN acoustic models for new speech corpora, languages, computational constraints, and language understanding task variants.

Our work systematically explores several dimensions of DNN design. We study the role of model size, and the interaction between model size and the number of network layers, addressing questions like how many layers are useful for conversational LVCSR, or for a given number of parameters, is it better to have wider or deeper networks? We study network architecture: can convolutional networks improve performance and obviate the need for complex feature extraction? And we examine a number of other parameters of DNN training, including different training loss functions and different techniques for reducing over-fitting.

We perform our DNN experiments on two corpora, first examining the standard Switchboard corpus. We analyze the effect of DNN size on task performance, and find that although there are 300 hours of training data we can cause DNNs to over-fit on this task by increasing DNN model size. We then investigate several techniques to reduce over-fitting including the popular dropout regularization technique. We next analyze neural network architecture choices by comparing deep convolutional neural networks (DCNNs), deep locally untied neural networks (DLUNNs), and standard DNNs. This comparison also evaluates alternative input features since convolutional approaches rely on input features with meaningful time and frequency dimensions.

To explore DNN performance with fewer constraints imposed by over-fitting, we next build a baseline LVCSR system by combining the Switchboard and Fisher corpora. This results in roughly 2100 hours of training data and represents one of the largest collections of conversational speech available for academic research. This larger corpus allows us to explore performance of much larger DNN models, up to ten times larger than those typically used for LVCSR. Using this larger corpus we also evaluate the impact of optimization algorithm choice, and the number of hidden layers used in a DNN with a fixed number of total free parameters. We analyze our results not only in terms of final task performance, but also compare sub-components of task performance across models.

Section 2 describes the neural network architectures and optimization algorithms evaluated in this paper. Section 3 presents our experiments on the Switchboard corpus, focusing on regularization and network dense versus convolutional architectural choices. We then present experiments on the combined Switchboard and Fisher corpora in Section 5, exploring the performance of larger and deeper DNN architectures, before concluding in Section 7.

2. Neural network computations

To address the stated research questions we employ three different classes of neural network architecture. Each architecture amounts to a different set of equations to convert input features into a predicted distribution over output classes. We describe here the specifics of each architecture, along with the loss function and optimization algorithms we use.

All of our experiments utilize the cross entropy classification loss function. For some experiments we apply regularization techniques in addition to the cross entropy loss function to improve generalization performance. The cross entropy loss function does not consider each utterance in its entirety. Instead it is defined over individual samples of acoustic input x and senone label y. The cross entropy objective function for a single training pair (x, y) is,

$$-\sum_{k=1}^{K} 1\{y = k\} \log \hat{y}_k, \tag{1}$$

where K is the number of output classes, and $\hat{y}_k$ is the probability that the model assigns to the input example taking on label k.

2.1. Deep neural network computations

A DNN is a series of fully connected hidden layers which transform an input vector x into a probability distribution $\hat{y}$ to estimate the output class. The DNN thus acts as a function approximator for the conditional distribution $p(y|x)$. A DNN parametrizes this function using L layers, a series of hidden layers followed by an output layer. Fig. 1 shows an example DNN.

Each layer has a weight matrix W and bias vector b. We compute vector h^1 of first layer activations of a DNN using,

$$h^{(1)}(x) = \sigma\left(W^{(1)T} x + b^{(1)}\right), \tag{2}$$

where $W^{(1)}$ and $b^{(1)}$ are the weight matrix and bias vectors respectively for the first hidden layer. In this formulation each column of the matrix $W^{(1)}$ corresponds to the weights for a single hidden unit of the first hidden layer. Because the DNN is fully connected, any real-valued matrix W forms a valid weight matrix. If we instead choose to impose partial connectivity, we are effectively constraining certain entries in W to be 0.

Subsequent hidden layers compute their hidden activation vector $h^{(i)}$ using the hidden activations of the previous layer $h^{(i-1)}$,

$$h^{(i)}(x) = \sigma\left(W^{(i)T} h^{(i-1)} + b^{(i)}\right). \tag{3}$$

In all hidden layers we apply a point-wise nonlinearity function $\sigma(z)$ as part of the hidden layer computation. Traditional approaches to neural networks typically use a sigmoidal function. However, in this work we use rectified linear units which were recently shown to lead to better performance in hybrid speech recognition as well as other DNN classification tasks (Dahl et al., 2013; Maas et al., 2013; Zeiler et al., 2013). The rectifier nonlinearity is defined as $\sigma(z) = \max(z, 0)$.

The final layer of the DNN must output a properly formed probability distribution over the possible output categories. To do this, the final layer of the DNN uses the *softmax* nonlinearity. Using the softmax nonlinearity we obtain the output vector $\hat{y}$ which is a well-formed probability distribution over the N output classes. This distribution can then be used in the loss function stated in Equation (1), or other loss functions. This DNN formulation is fairly

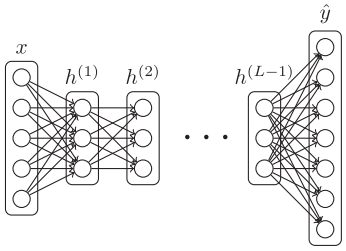

Fig. 1. A DNN with 5-dimensional input, 3-dimensional hidden layers, and 7-dimensional output. Each hidden layer is fully connected to the to the previous and subsequent layer.

standard when compared to work in the speech recognition community. The choice of rectifier nonlinearities is a new one, but their benefit has been reproduced by several research groups.

2.2. Deep convolutional neural networks

The fully-connected DNN architecture presented thus far serves as the primary neural network acoustic modeling choice for modern speech recognition tasks. In contrast, neural networks for computer vision tasks are often deep convolutional neural networks (DCNNs) which exploit spatial relationships in input data (Krizhevsky et al., 2012; LeCun et al., 1998). DCNNs follow a convolutional layer with a pooling layer to hard-code invariance to slight shifts in time and frequency. Like fully connected neural network acoustic models, the idea of using localized time–frequency regions for speech recognition was introduced over 20 years ago (Waibel et al., 1989). We evaluate DCNN acoustic models consistent with recent work in convolutional acoustic modeling (Sainath et al., 2014).

The initial layers in a DCNN use convolutional layers in place of the standard fully-connected layers present in DNNs. In a convolutional model, we restrict the total number of network parameters by using hidden units which connect to only a small, localized region of the input. These localized hidden units are applied at many different spatial locations to obtain hidden layer representations for the entire input.

Fig. 2 shows a convolutional hidden layer connected to input features with time and frequency axes. A single weight matrix W_1 connects to a 3×3 region of the input and we compute a hidden unit activation value using the rectifier nonlinearity. We apply this same procedure at all possible locations of the input, moving one step at a time across the input in both dimensions. This process produces a *feature map* $h^{(1,1)}$ which is the hidden activation values for W_1 at each location of the input. The feature map itself has meaningful time and frequency axes because we preserve these dimensions as we convolve across the input to compute hidden unit activations.

Following the convolutional layer, we apply a *pooling* operation. Pooling acts as a down-sampling step, and hard-codes invariance to slight translations in the input. Like the localized windows used in the convolutional layer, the pooling layer connects to a contiguous, localized region of its input – the feature map produced by a convolutional hidden layer. The pooling layer does not have overlapping regions. We apply this pooling function to local regions in each feature map. Recall that a feature map contains the hidden unit activations for only a single hidden unit. We are thus using pooling to select activation values for each hidden unit separately, and not forcing different hidden units to compete with one another. In our work, we use *max pooling* which applies a max function to the set of inputs in a single pooling region. Max pooling is a common choice of pooling function for neural networks in both computer vision and acoustic modeling tasks (Abdel-Hamid et al., 2012; Lee et al., 2009; Sainath et al., 2014). The most widely used alternative to max pooling replaces the max function with an averaging function. Results with max pooling and average pooling are often comparable.

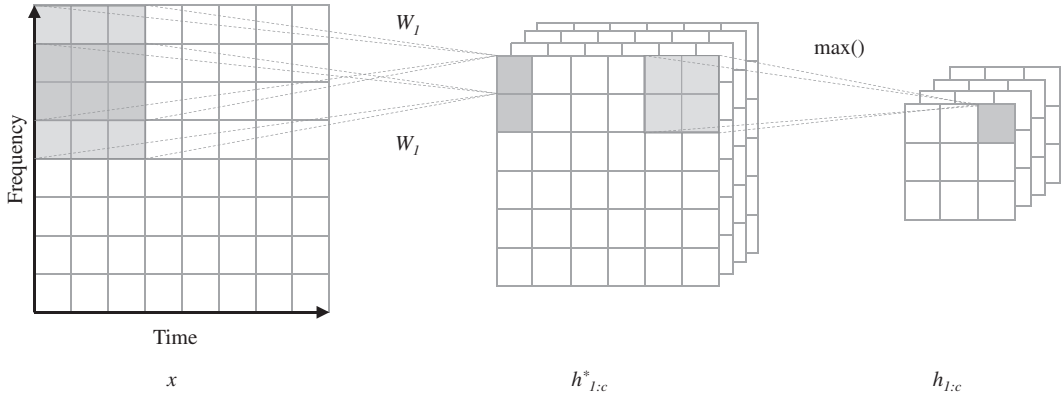

Fig. 2. Convolution and pooling first layer architecture. Here the filter size is 5×5, and the pooling dimension is 3×3. Pooling regions are non-overlapping. Note that the 5×5 filters applied to each position in the convolution step are constrained to be the same. For max-pooling, the maximum value in each 3×3 grid is extracted.

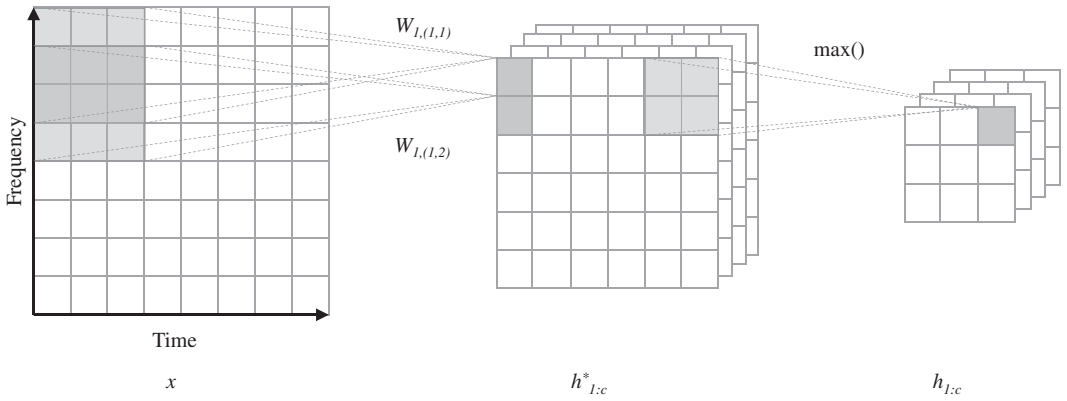

Fig. 3. Locally connected untied first layer architecture. Here the filter size is 5×5, and the pooling dimension is 3×3. Pooling regions are non-overlapping. Unlike the convolutional layer shown in Fig. 2, the network learns a unique 5×5 set of weights at each location. The max pooling layer otherwise behaves identically to the pooling layer in a convolutional architecture.

2.3. Deep local untied neural networks

DCNNs combine two architectural ideas simultaneously – locally-connected hidden units and sharing weights across multiple hidden units. We need not apply both of these architectural ideas simultaneously. In a *deep local untied neural network* (DLUNN) we again utilize locally-connected hidden units but do not share weights at different regions of the input. Fig. 3 shows an example DLUNN architecture, which differs only from a DCNN architecture by using different weights at each location of the first hidden layer. When applying a local untied hidden layer to Mel-spectrum time–frequency input features the hidden units can process different frequency ranges using different hidden units. This allows the network to learn slight variations that may occur when a feature occurs at a lower frequency versus a higher frequency.

In DLUNNs, the architecture is the same as in the convolutional network, except that filters applied to different regions of the input are not constrained to be the same. Thus untied neural networks can be thought of as convolutional neural networks using locally connected computations and without weight-sharing. This results in a large increase in the number of parameters for the untied layers relative to DCNNs. Following each locally untied layer we apply a max pooling layer which behaves identically to the pooling layers in our DCNN architecture. Grouping units together with a max pooling function often results in hidden weights being similar such that the post-pooling activations are an invariant feature which detects a similar time–frequency pattern at different regions of the input.

2.4. Optimization algorithms

Choosing optimization algorithms for DNN training is difficult because, given the non-convex problems we need to optimize, we usually can't say in general whether one algorithm is better than another. In this work we consider two of the most popular stochastic gradient techniques for our neural network training.

The first optimization algorithm we consider is stochastic gradient with classical momentum (CM) (Plaut, 1986; Rumelhart et al., 1986). This technique is probably the most standard optimization algorithm choice in modern neural network research. To minimize a cost function $f(\theta)$ classical momentum updates amount to,

$$v_t = \mu v_{t-1} - \varepsilon \nabla f\left(\theta_{t-1}\right) \tag{4}$$

$$\theta_t = \theta_{t-1} + v_t, \tag{5}$$

where v_t denotes the accumulated gradient update, or *velocity*, $\varepsilon > 0$ is the learning rate, and the momentum constant $\mu \in [0, 1]$ governs how we accumulate the velocity vector over time. By setting μ close to one, one can expect to accumulate the gradient information across a larger set of past updates.

Recently the Nesterov's accelerated gradient (NAG) (Nesterov, 1983) technique was found to address some of the issues encountered when training neural networks with CM. NAG accumulates past gradients using an alternative update equation that finds a better objective function value with less sensitivity to optimization algorithm hyper-parameters on some neural network tasks. The NAG update rule is defined as,

$$v_t = \mu_{t-1} v_{t-1} - \varepsilon_{t-1} \nabla f\left(\theta_{t-1} + \mu_{t-1} v_{t-1}\right) \tag{6}$$

$$\theta_t = \theta_{t-1} + v_t. \tag{7}$$

Intuitively, this method avoids potential fluctuation in the optimization by looking ahead to the gradient along the update direction. For a more detailed explanation of the intuition underlying NAG optimization for neural network tasks see Sutskever (2013).

3. Switchboard 300 hour corpus

We first carry out LVCSR experiments on the 300 hour Switchboard conversational telephone speech corpus (LDC97S62). The baseline GMM system and forced alignments are created using the Kaldi open-source toolkit (Povey et al., 2011).[1] We build a baseline GMM system by following the s5b Kaldi recipe and we defer to a previous work for details (Vesel et al., 2013). The baseline recognizer has 8986 sub-phone states and 200k Gaussians. The DNN is trained to estimate state likelihoods which are then used in a standard hybrid HMM/DNN setup. Input features for the DNNs are MFCCs with a context of ±10 frames. Per-speaker CMVN is applied and speaker adaptation is done using fMLLR. For recognition evaluation, we report on a test set Eval2000 consisting of both the Switchboard and CallHome subsets of the HUB5 2000 data (LDC2002S09). We also report results on a subset of the training set consisting of 5000 utterances.

3.1. Varying DNN model size

We first experiment with perhaps the most direct approach to improving performance with DNNs – making DNNs larger by adding hidden units. Increasing the number of parameters in a DNN directly increases the representational capacity of the model. Indeed, this representational scalability drives much of the modern interest in applying DNNs to large datasets which might easily saturate other types of models. Many existing experiments with DNN acoustic models focus on introducing architecture or loss function variants to further specialize DNNs for speech tasks. We instead ask the question of whether model size alone can drive significant improvements in overall system performance. We additionally experiment with using a larger context window of frames as a DNN input as this should also serve as a direct path to improving the frame classification performance of DNNs.

3.1.1. Experiments

We explore three different model sizes by varying the total number of parameters in the network. The number of hidden layers is fixed to five, so altering the total number of parameters affects the number of hidden units in each layer. All hidden layers in a single network have the same number of hidden units. The hidden layer sizes are 2048, 3953 and 5984 which respectively yield models with approximately 36 million (M), 100M and 200M parameters. There are 8986 output classes which results in the output layer being the largest single layer in any of our networks. In DNNs of the size typically studied in the literature this output layer often consumes a majority of the total parameters in the network. For example in our 36M parameter model the output layer comprises 51% of all parameters. In contrast, the output layer in our 200M model is only 6% of total parameters. Many output classes occur rarely so devoting a large fraction of network parameters to class-specific modeling may be wasteful. Previous works explore factoring the output layer to increase the relative number of shared parameters (Liao et al., 2013; Sainath et al., 2013), but this effect occurs naturally by substantially increasing network size. For our larger models we experiment with the standard input of ±10 context frames and additionally models trained with ±20 context frames.

[1] http://kaldi.sf.net.

Table 1
Results for DNN systems in terms of frame-wise error metrics on the development set as well as word error rates on the training set and Hub5 2000 evaluation sets.

Model size	Layer size	Context	Dev cross ent	Dev acc(%)	Train WER	SWBD WER	CH WER	EV WER
GMM baseline	N/A	±0	N/A	N/A	24.93	21.7	36.1	29.0
36M	2048	±10	1.23	66.20	17.52	15.1	27.1	21.2
100M	3953	±10	0.77	78.56	13.66	14.5	27.0	20.8
100M	3953	±20	0.50	85.58	12.31	14.9	27.7	21.4
200M	5984	±10	0.51	86.06	11.56	15.0	26.8	20.9
200M	5984	±20	0.26	93.05	10.09	15.4	28.5	22.0

The Hub5 set (EV) contains the Switchboard (SWBD) and CallHome (CH) evaluation subsets. We also include word error rates for the Fisher corpus development set (FSH) for cross-corpus comparison. Frame-wise error metrics were evaluated on 1.7M frames held out from the training set. DNN models differ only by their total number of parameters. All DNNs have 5 hidden layers with either 2048 hidden units (36M parameters), 3953 hidden units (100M parameters), or 5984 hidden units (200M parameters).

All models use hidden units with the rectified linear nonlinearity. For optimization, we use Nesterov's accelerated gradient with a smooth initial momentum schedule which we clamp to a maximum of 0.95 (Sutskever et al., 2013). The stochastic updates are on mini-batches of 512 examples. After each epoch, or full pass through the data, we anneal the learning rate by half. Training is stopped after improvement in the cross entropy objective evaluated on held out development set falls below a small tolerance threshold.

In order to efficiently train models of the size mentioned above, we distribute the model and computation across several GPUs using the distributed neural network infrastructure proposed by Coates et al. (2013). Our GPU cluster and distributed training software is capable of training up to 10 billion parameter DNNs. We restrict our attention to models in the 30M–200M parameter range. In preliminary experiments we found that DNNs with 200M parameters are representative of DNNs with over one billion parameters for this task. We train models for this paper in a model-parallel fashion by distributing the parameters across four GPUs. A single pass through the training set for a 200M parameter DNN takes approximately 1.5 days. Table 1 shows frame-level and WER evaluations of acoustic models of varying size compared against our baseline GMM recognizer.

3.1.2. Results

Table 1 shows results for DNNs of varying size and varying amounts of input context. We find that substantially increasing DNN size shows clear improvements in frame-level metrics. Our 200M parameter DNN halves the development set cross entropy cost of the smaller 36M parameter DNN – a substantial reduction. For each increase in DNN model size there is approximately a 10% absolute increase in frame classification accuracy. Frame-level metrics are further improved by using larger context windows. In all cases a model trained with larger context window outperforms its smaller context counterpart. Our best overall model in terms of frame-level metrics is a 200M parameter DNN with context window of ±20 frames.

However, frame-level performance is not always a good proxy for WER performance of a final system. We evaluate WER on a subset of the training data as well as the final evaluation sets. Large DNN acoustic models substantially reduce WER on the training set. Indeed, our results suggest that further training set WER reductions are possible by continuing to increase DNN model size. However, the gains we observe on the training set in WER do not translate to large performance gains on the evaluation sets. While there is a small benefit of using models larger than the 36M DNN baseline size, building models larger than 100M parameters does not prove beneficial for this task.

3.1.3. Discussion

To better understand the dynamics of training large DNN acoustic models, we plot training and evaluation WER performance during DNN training. Fig. 4 shows WER performance for our 100M and 200M parameter DNNs after each epoch of cross entropy training. We find that training WER reduces fairly dramatically at first and then continues to decrease at a slower but still meaningful rate. In contrast, nearly all of our evaluation set performance is realized within the first few epochs of training. This has two important practical implications for large DNN training for speech recognition. First, large acoustic models are not beneficial but do not exhibit a strong over-fitting effect where evaluation set performance improves for awhile before becoming increasingly worse. Second, it may be possible to

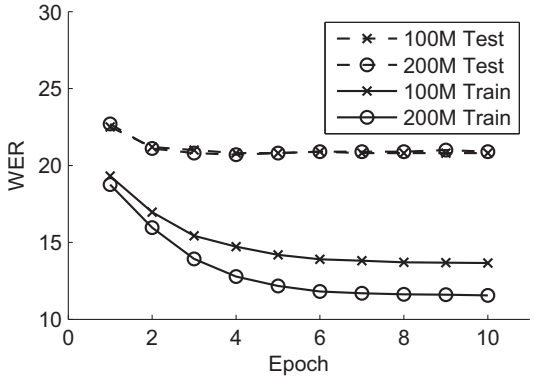

Fig. 4. Train and test set WER as a function of training epoch for systems with DNN acoustic models of varying size. Each epoch is a single complete pass through the training set. Although the training error rate is substantially lower for large models, there is no gain in test set performance.

utilize large DNNs without prohibitively long training times by utilizing our finding that most performance comes from the first few epochs, even with models at our scale. Finally, although increasing context window size improves all training set metrics, those gains do not translate to improved test set performance. It seems that increasing context window size provides an easy path to better fitting the training function, but does not result in the DNN learning a meaningful, generalizable function.

3.2. Dropout regularization

Dropout is a recently-introduced technique to prevent over-fitting during DNN training (Hinton et al., 2012). The dropout technique randomly masks out hidden unit activations during training, which prevents co-adaptation of hidden units. For each example observed during training, each unit has its activation set to zero with probability $p \in [0, 0.5]$. Several experiments demonstrate dropout as a good regularization technique for tasks in computer vision and natural language processing (Krizhevsky et al., 2012; Wager et al., 2013. Dahl et al. (2013) found a reduction in WER when using dropout on a 10M parameter DNN acoustic model for a 50 hour broadcast news LVCSR task. Dropout additionally yielded performance gains for convolutional neural networks with less than 10M parameters on both 50 and 400 hour broadcast news LVCSR tasks (Sainath et al., 2013). While networks which employ dropout during training were found effective in these studies, the authors did not perform control experiments to measure the impact of dropout alone. We directly compare a baseline DNN to a DNN of the same architecture trained with dropout. This experiment tests whether dropout regularization can mitigate the poor generalization performance of large DNNs observed in Section 3.1.

3.2.1. Experiments

We train DNN acoustic models with dropout to compare generalization WER performance against that of the DNNs presented in Section 3. The probability of dropout p is a hyper-parameter of DNN training. In preliminary experiments we found setting $p = 0.1$ to yield the best generalization performance after evaluating several possible values, $p \in \{0.01, 0.1, 0.25, 0.5\}$. The DNNs presented with dropout training otherwise follow our same training and evaluation protocol used thus far, and are built using the same forced alignments from our baseline HMM-GMM system.

3.2.2. Results

Table 2 shows the test set performance of DNN acoustic models of varying size trained with dropout. DNNs trained with dropout improve over the baseline model for all acoustic model sizes we evaluate. The improvement is a consistent 0.2% to 0.4% reduction in absolute WER on the test set. While beneficial, dropout seems insufficient to fully harness the representational capacity of our largest models. Additionally, we note that hyper-parameter selection was critical to finding any gain when using dropout. With a poor setting of the dropout probability p preliminary experiments found no gain and often worse results from training with dropout.

Table 2
Results for DNN systems trained with dropout regularization (DO) and early realignment (ER) to improve generalization performance.

Model	SWBD	CH	EV
GMM Baseline	21.7	36.1	29.0
2048 Layer (36M)	15.1	27.1	21.2
2048 Layer (36M) DO	14.7	26.7	20.8
3953 Layer (100M)	14.7	26.7	20.7
3953 Layer (100M) DO	14.6	26.3	20.5
3953 Layer (100M) ER2	14.3	26.0	20.2
3953 Layer (100M) ER5	14.5	26.4	20.5
5984 Layer (200M)	15.0	26.9	21.0
5984 Layer (200M) DO	14.9	26.3	20.7

We build models with early realignment by starting realignment after each epoch starting after epoch two (ER2) and epoch five (ER5). Word error rates are reported on the combined Hub5 test set (EV) which contains Switchboard (SWBD) and CallHome (CH) evaluation subsets. DNN model sizes are shown in terms of hidden layer size and millions of total parameters (e.g. 100M).

3.3. Early stopping

Early stopping is a regularization technique for neural networks which halts loss function optimization before completely converging to the lowest possible function value. We evaluate early stopping as another standard DNN regularization technique which may improve the generalization performance of large DNN acoustic models. Previous work by Caruana et al. (2000) found that early stopping training of networks with large capacity produces generalization performance on par with or better than the generalization of a smaller network. Further, this work found that, when using back-propagation for optimization, early in training a large capacity network behaves similarly to a smaller capacity network. Finally, early stopping as a regularization technique is similar to an ℓ_2 weight norm penalty, another standard approach to regularization of neural network training.

3.3.1. Results

By analyzing the training and test WER curves in Fig. 4 we can observe the best-case performance of an early stopping approach to improving generalization. If we select the lowest test set WER the system achieves during DNN optimization, the 200M parameter DNN achieves 20.7% WER on the EV subset – only 0.1% better than the 100M parameter baseline DNN system. This early stopped 200M model achieves only a 0.5% absolute WER reduction over the much smaller 36M parameter DNN. This suggests that early stopping is beneficial, but perhaps insufficient to yield the full possible benefits of large DNN acoustic models.

3.4. Early realignment

Acoustic model training data is labeled via a forced alignment of the word-level transcriptions using the current partially-trained system. We can think of the alignment of state labels to audio features as latent variables which we approximate in order to train the acoustic model. We test whether re-estimating the alignment *during* training using the partially-trained DNN leads to improved generalization performance.

Neural networks exhibit interesting dynamics during optimization. Work on early stopping found that networks with high capacity exhibit behavior similar to smaller, limited capacity networks in early phases of optimization (Caruana et al., 2000). Combining this finding with the generally smooth functional form of DNN hidden and output units suggests that early in training a large capacity DNN may fit a smooth output function which ignores some of the label noise introduced by imperfect forced alignments. Of course, a large enough DNN should completely fit the corruptions present in training data as optimization converges. Studies on the learning dynamics of DNNs for hierarchical categorization tasks additionally suggest that coarse, high-level output classes are fit first during training optimization (Saxe et al., 2013).

Baseline LVCSR systems using GMM acoustic models realign several times during training to iteratively improve. While iterative realignments have been helpful in improving system performance in single-layer ANN-HMM hybrid models (Bourlard and Morgan, 1993), realignment is typically not used with large DNN acoustic models because of

the long training times of DNNs. However, realignment using a fully trained DNN acoustic model often can produce a small reduction in final system WER (Hinton et al., 2012).

We evaluate *early realignment* which generates a new forced alignment early in DNN optimization and then continues training on the new set of labels. Because large capacity DNNs begin accurately predicting labels much earlier in training, early realignment may save days of training time. Further, we hypothesize that a less fully converged network can remove some label distortions while a more completely trained DNN may already be fitting to the corrupt labels given by an imperfect alignment.

3.4.1. Experiments

We begin by training an initial DNN using the same HMM-GMM forced alignments and non-regularized training procedures presented thus far. After training the DNN using the initial HMM-GMM alignments for a fixed number of epochs, we use our new HMM-DNN system to generate a new forced alignment for the entire training set. DNN training then proceeds using the same DNN weights but the newly-generated training set labels. As in our regularization experiments, we hold the rest of our DNN training and evaluation procedures fixed to directly measure the impact of early realignment training. We train 100M parameter five hidden layer DNNs and build models by realigning after either two or five epochs.

In preliminary experiments we found that realignment after each epoch was too disruptive to DNN training and resulted in low quality DNN models. Similarly, we found that starting from a fresh, randomly initialized DNN after realignment performed worse than continuing training from the DNN weights used to generate the realignment. We found it important to reset the stochastic gradient learning rate to its initial value after realignment occurs. Without doing so, our annealing schedule sets the learning rate too low for the optimization procedure to fully adjust to the newly-introduced labels. In a control experiment, we found that resetting the learning rate alone, without realignment, does not improve system performance.

3.4.2. Results

Table 2 compares the final test set performance of DNNs trained with early realignment to a baseline model as well as DNNs trained with dropout regularization. Realignment after five epochs is beneficial compared to the baseline DNN system, but slightly worse than a system which realigns after two epochs of training. Early realignment leads to better WER performance than all models we evaluated, trained with dropout and early stopping. This makes early realignment the overall best technique we evaluated on the Switchboard corpus. We note that only *early* realignment outperforms dropout regularization – a DNN trained with realignment after five epochs performs comparably to a DNN of the same size trained with dropout.

3.4.3. Discussion

Fig. 5 shows training and test WER curves for 100M parameter DNN acoustic models trained with early realignment and a baseline DNN with no realignment. We note that just after realignment both train and test WER increase

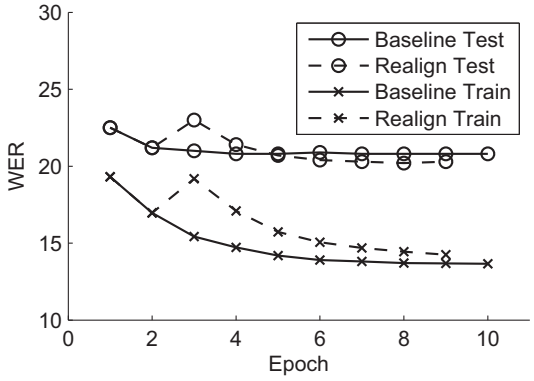

Fig. 5. WER as a function of DNN training epoch for systems with DNN acoustic models trained with and without label realignment after epoch 2. A DNN which re-generates its training labels with a forced alignment early during optimization generalizes much better to test data than a DNN which converges to the original labels.

briefly. This is not surprising as realignment substantially changes the distribution of training examples. The DNN trained with realignment trains for three epochs following realignment before it begins to outperform the baseline DNN system.

We can quantify how much the labeling from realignment differs from the original labeling by computing the fraction of labels changed. In early realignment 16.4% of labels are changed by realignment while only 10% of labels are changed when we realign with the DNN trained for five epochs. This finding matches our intuition that as a large capacity DNN trains it converges to fit the corrupted training samples extremely well. Thus when we realign the training data with a fully trained large capacity DNN the previously observed labels are reproduced nearly perfectly. Realigning with a DNN earlier in optimization mimics realigning with a higher bias model which relabels the training set with a smoother approximate function. Taken together, our results suggest that early realignment leverages the high bias characteristics of the initial phases of DNN training to reduce WER while requiring minimal additional training time.

Early realignment also shows a huge benefit to training time compared to traditional realignment. The DNN trained with realignment after epoch five must train an additional three epochs, for a total of eight, before it can match the performance of a DNN trained with early realignment. For DNNs of the scale we use, this translates to several days of compute time. The training time and WER reduction of DNNs with early realignment comes with a cost of implementing and performing realignment, which is of course not a standard DNN training technique. Overall, we conclude that early realignment is an effective technique to improve performance of DNN acoustic models with minimal additional training time.

4. Comparing DNNs, DCNNs, and DLUNNs on switchboard

The experiments thus far modify DNN training by adding various forms of regularization. We now experiment with alternative neural network architectures – deep convolutional neural networks (DCNNs) and deep local untied neural networks (DLUNNs).

4.1. Experiments

We trained DCNN and DLUNN acoustic models using the same Switchboard training data as used for our DNN acoustic model experiments to facilitate direct comparisons across architectures. We evaluate filter bank features in addition to the fMLLR features used in DNN training because filter bank features have meaningful spectro-temporal dimensions for local receptive field computations. All models have five hidden layers and were trained using Nesterov's accelerated gradient with a smoothly increasing momentum schedule capped at 0.95 and a step size of 0.01, halving the step size after each epoch.

For our DCNN and DLUNN acoustic models we chose a receptive field of 9×9 and non-overlapping pooling regions of dimension 1×3 (time by frequency). Our models with two convolutional layers have the same first layer filter and pooling sizes. The second layer uses a filter size of 3×3 and does not use pooling. These parameters were selected using results from preliminary experiments as well as results from a previous work (Sainath et al., 2013).

In the DCNNs one convolutional layer was used followed by four densely connected layers with equal number of hidden units, and similarly for the DLUNNs. Map depth and number of hidden units were selected such that all models have approximately 36M parameters. For DCNNs, the convolutional first layer has a map depth of 128 applied to an input with ±10 frame context. The following dense hidden layers each have 1240 hidden units. Our 2 convolutional layer DCNN uses 128 feature maps in both convolutional layers and 3 dense layers with 1240 hidden units each. All DLUNNs use 108 filters at each location in the first layer, and 4 hidden layers each with 1240 hidden units.

The filter bank and fMLLR features are both 40-dimensional. We ran initial experiments convolving filters along frequency only, pooling along both frequency and time, and overlapping pooling regions, but did not find that these settings gave better performance. We ran experiments with a context window of ±20 frames but found results to be worse than results obtained with a context window of ±10 frames, so we report only the ±10 frame context results.

4.2. Results

Table 3 shows the frame-level and final system performance results for acoustic models built from DNNs, DCNNs, and DLUNNs. When using filter bank features, DCNNs and DLUNNs both achieve improvements over DNNs. DCNN

Table 3
Performance comparison of DNNs, deep convolutional neural networks (DCNNs), and deep local untied neural networks (DLUNNs).

Model	Features	Acc (%)	SWBD WER	CH WER	EV WER
GMM	fMLLR	N/A	21.7	36.1	29.0
DNN	fMLLR	60.8	14.9	27.4	21.2
DNN	FBank	51.7	16.5	31.6	24.1
DCNN	fMLLR	59.3	15.8	28.3	22.0
DCNN	FBank	53.0	15.8	28.7	22.3
DCNN	fMLLR-P	59.0	15.9	28.6	22.4
DCNN	FBank-P	50.7	17.2	32.1	24.7
DCNN2	fMLLR	58.8	15.9	28.3	22.2
DCNN2	FBank	53.0	15.6	28.3	22.1
DLUNN	fMLLR	61.2	15.2	27.4	21.3
DLUNN	FBank	53.0	16.1	29.3	22.8

We evaluate convolutional models with one layer of convolution (DCNN) and two layers of convolution (DCNN2). We compare models trained with fMLLR features and filter bank (FBank) features. Note that a context window of fMLLR features has a temporal dimension but no meaningful frequency dimension whereas FBank features have meaningful time–frequency axes. As an additional control we train a DCNN on features which are randomly permuted to remove meaningful coherence in both the time and frequency axes (FBank-P and fMLLR-P). We report performance on both the Hub5 Eval2000 test set (EV) which contains Switchboard (SWBD) and CallHome (CH) evaluation subsets.

models narrowly outperform DLUNN models. For locally connected acoustic models it appears that the constraint of tied weights in convolutional models is advantageous as compared to allowing a different set of localized receptive fields to be learned at different time–frequency regions of the input.

DLUNNs outperform DCNNs in experiments with fMLLR features. Indeed, the DLUNN performs about as well as the DNN. The DCNN is harmed by the lack of meaningful relationships along the frequency dimension of the input features, whereas the more flexible architecture of the DLUNN is able to learn useful first layer parameters. We also note that our fMLLR features yield much better performance for all models as compared with models trained on filter bank features.

In order to examine how much benefit using DCNNs to leverage local correlations in the acoustic signal yields, we ran control experiments with filter bank features randomly permuted along both the frequency and time axes. The results show that while this harms performance the convolutional architecture can still obtain fairly competitive word error rates. This control experiment confirms that locally connected models do indeed leverage localized properties of the input features to achieve improved performance.

4.3. Discussion

While DCNN and DLUNN models are promising as compared to DNN models on filter bank features, our results with filter bank features are overall worse than results from models utilizing fMLLR features.

Note that the filter bank features we used are fairly simple as compared to our fMLLR features as the filter bank features do not contain significant post-processing for speaker adaptation. While performing such feature transformations may give improved performance, they call into question the initial motivation for using DCNNs to automatically discover invariance to gender, speaker and time–frequency distortions. The fMLLR features we compare against include much higher amounts of specialized post-processing, which appears beneficial for all neural network architectures we evaluated. This confirms recent results from previous work, which found that DCNNs alone are not typically superior to DNNs but can complement a DNN acoustic model when both are used together, or achieve competitive results when increased amounts of post-processing are applied to filter bank features (Sainath et al., 2014). In summary, we conclude that DCNNs and DLUNNs are not sufficient to replace DNNs as a default, reliable choice for acoustic modeling network architecture. We additionally conclude that DLUNNs warrant further investigation as alternatives to DCNNs for acoustic modeling tasks.

5. Combined large corpus

On the Switchboard 300 hour corpus we observed limited benefits from increasing DNN model size for acoustic modeling, even with a variety of techniques to improve generalization performance. We next explore DNN performance

using a substantially larger training corpus. This set of experiments explores how we expect DNN acoustic models to behave when training set size is not a limiting factor. In this setting, over-fitting with large DNNs should be less of a problem and we can more thoroughly explore architecture choices in large DNNs rather than regularization techniques to reduce over-fitting and improve generalization with a small training corpus.

5.1. Baseline HMM system

To maximize the amount of training data for a conversational speech transcription task, we combine the Switchboard corpus with the larger Fisher corpus (Cieri et al., 2004). The Fisher corpus contains approximately 2000 hours of training data, but has transcriptions which are slightly less accurate than those of the Switchboard corpus.

Our baseline GMM acoustic model was trained on features that are obtained by splicing together 7 frames (3 on each side of the current frame) of 13-dimensional MFCCs (C0-C12) and projecting down to 40 dimensions using linear discriminant analysis (LDA). The MFCCs are normalized to have zero mean per speaker.[2] After obtaining the features with LDA, we also use a single semi-tied covariance (STC) transform on the features. Moreover, speaker adaptive training (SAT) is done using a single feature-space maximum likelihood linear regression (fMLLR) transform estimated per speaker. The models trained on the full combined Fisher + Switchboard training set contain 8725 tied triphone states and 3.2M Gaussians.

The language model in our baseline system is trained on the combination of the Fisher transcripts and the Switchboard Mississippi State transcripts. Kneser–Ney smoothing was applied to fine-tune the back-off probabilities to minimize the perplexity on a held out set of 10K transcript sentences from Fisher transcripts. In preliminary experiments we interpolated the transcript-derived language model with a language model built from a large collection of web page text, but found no gains as compared with using the transcript-derived language model alone.

We use two evaluation sets for all experiments on this corpus. First, we use the same Hub5'00 (Eval2000) corpus used to evaluate systems on the Switchboard 300 hr task. This evaluation set serves as a reference point to compare systems built on our combined corpus to those trained on Switchboard alone. Second, we use the RT-03 evaluation set which is more frequently used in the literature to evaluate Fisher-trained systems. Performance of the baseline HMM-GMM system is shown in Tables 4 and 5.[3]

5.2. Optimization algorithm choice

To avoid exhaustively searching over all DNN architecture and training parameters simultaneously, we first establish the impact of optimization algorithm choice while holding the DNN architecture fixed. We train networks with the two optimization algorithms described in Section 2.4 to determine which optimization algorithm to use in the rest of the experiments on this corpus.

5.2.1. Experiments

We train several DNNs with five hidden layers, where each layer has 2048 hidden units. This results in DNNs with roughly 36M total free parameters, which is a typical size for acoustic models used for conversational speech transcription in the research literature. For both the classical momentum and Nesterov's accelerated gradient optimization techniques the two key hyper-parameters are the initial learning rate ε and the maximum momentum μ_{max}. In all cases we decrease the learning rate by a factor of 2 every 200,000 iterations. This learning rate annealing was chosen after preliminary experiments, and overall performance does not appear to be significantly affected by annealing schedule. It is more common to anneal the learning rate after each pass through the dataset. Because our dataset is quite large we found that annealing only after each epoch leads to much slower convergence to a good optimization solution.

5.2.2. Results

Table 4 shows both WER performance and classification accuracy of DNN-based ASR systems with various optimization algorithm settings. We first evaluate the effect of optimization algorithm choice. We evaluated DNNs with

[2] This is done strictly for each individual speaker with our commit r4258 to the Kaldi recognizer. We found this to work slightly better than normalizing on a per conversation-side basis.

[3] The implementation of our baseline HMM-GMM system is available in the Kaldi project repository as example recipe fisher_swbd (revision: r4340).

Table 4
Results for DNNs of the same architecture trained with varying optimization algorithms.

Optimizer	μ_{max}	Acc (%)	SWBD WER	CH WER	EV WER	RT03 WER
GMM	N/A	N/A	21.9	31.9	26.9	39.5
CM	0.90	52.51	18.3	27.3	22.8	39.0
CM	0.95	54.20	17.1	25.6	21.4	38.1
CM	0.99	55.26	16.3	24.8	20.6	37.5
NAG	0.90	53.18	18.0	26.7	22.3	38.5
NAG	0.95	54.27	17.2	25.8	21.5	39.6
NAG	0.99	55.39	16.3	24.7	20.6	37.4

Primarily we compare stochastic gradient using classical momentum (CM) and Nesterov's accelerated gradient (NAG). We additionally evaluate multiple settings for the maximum momentum (μ_{max}). The table contains results for only one learning rate ($\varepsilon = 0.01$) since it produces the best performance for all settings of optimization algorithm and momentum. We report performance on both the Hub5 Eval2000 test set (EV) which contains Switchboard (SWBD) and CallHome (CH) evaluation subsets. We also evaluate performance on the RT03 (RT03) Switchboard test set for comparison with Fisher corpus systems.

$\mu_{max} \in 0.9, 0.95, 0.99$ and $\varepsilon \in \{0.1, 0.01, 0.001\}$. For both optimization algorithms DNNs achieve the best performance by setting $\mu_{max} = 0.99$ and $\varepsilon = 0.01$.

In terms of frame level accuracy the NAG optimizer narrowly outperforms the CM optimizer, but WER performance across all evaluation sets are nearly identical. For both optimization algorithms a high value of μ_{max} is important for good performance. Note that most previous work in hybrid acoustic models use CM with $\mu_{max} = 0.90$, which does not appear to be optimal in our experiments. We also found that a larger initial learning rate was beneficial. We ran experiments using $\varepsilon \geq 0.05$ but do not report results because the DNNs diverged during the optimization process. Similarly, all models trained with $\varepsilon = 0.001$ had WER more than 1% absolute higher on the EV test set as compared to the same architecture trained with $\varepsilon = 0.01$. We thus omit the results for models trained with $\varepsilon = 0.001$ from our results table.

For the remainder of our experiments we use the NAG optimizer with $\mu_{max} = 0.99$ and $\varepsilon = 0.01$. These settings achieve the best performance overall in our initial experiments, and generally we have found the NAG optimizer to be somewhat more robust than the CM optimizer in producing good parameter solutions.

5.3. Scaling total number of DNN parameters

We next evaluate the performance of DNNs as a function of the total number of model parameters while keeping network depth and optimization parameters fixed. This approach directly assesses the hypothesis of improving performance as a function of model size when there is sufficient training data available. We train DNNs with 5 hidden layers, and keep the number of hidden units constant across each hidden layer. Varying total free parameters thus corresponds to adding hidden units to each hidden layer. Table 5 shows the frame classification and WER performance of 5 hidden layer DNNs containing 36M, 100M, 200M, and 400M total free parameters.

Overall, the 400M parameter model performs best in terms of both frame classification and WER across all evaluation sets. Unlike with our smaller Switchboard training corpus experiments, increasing DNN model size does not lead to significant over-fitting problems in WER. However, the gain from increasing model size from 36M to 400M, more than a 10× increase, is somewhat limited. On the Eval2000 evaluation set we observe a 3.8% relative gain in WER from the 100M DNN as compared to the 36M DNN. When moving from the 100M DNN to the 200M DNN there is relative WER gain of 2.5%. Finally the model size increase from 200M to 400M total parameters yields a relative WER gain of 1%. There are clearly diminishing returns as we increase model size. The trend of diminishing relative gains in WER also occurs on the RT03 evaluation set, although relative gains on this evaluation set are somewhat smaller overall.

Frame classification rates on this corpus are much lower overall as compared with our Switchboard corpus DNNs. We believe that this corpus is more challenging due to more overall acoustic variation, and errors induced by quick transcriptions. Even our largest DNN leaves room for improvement in terms of frame classification. In Section 6 we explore more thoroughly the frame classification performance of the DNNs presented here.

Table 5
Results for DNNs of varying total model size and DNN depth.

# Params	Num. layers	Layer size	Acc (%)	SWBD WER	CH WER	EV WER	RT03 WER
GMM	N/A	N/A	N/A	21.9	31.9	26.9	39.5
36M	1	3,803	49.38	21.0	30.4	25.8	43.2
36M	3	2,480	54.78	17.0	25.8	21.4	38.2
36M	5	2,048	55.37	16.2	24.7	20.6	37.4
36M	7	1,797	54.99	16.3	24.7	20.7	37.3
100M	1	10,454	50.82	19.8	29.1	24.6	42.4
100M	3	4,940	56.02	16.3	24.8	20.6	37.3
100M	5	3,870	56.62	15.8	23.8	19.8	36.7
100M	7	3,309	56.59	15.7	23.8	19.8	36.4
200M	1	20,907	51.29	19.6	28.7	24.3	42.8
200M	3	7,739	56.58	16.0	24.0	20.1	37.0
200M	5	5,893	57.36	15.3	23.1	19.3	36.0
200M	7	4,974	57.28	15.3	23.3	19.3	36.2
400M	5	8,876	57.70	15.0	23.0	19.1	35.9

We report performance on both the Hub5 Eval2000 test set (EV) which contains Switchboard (SWBD) and CallHome (CH) evaluation subsets. We also evaluate performance on the RT03 (RT03) Switchboard test set for comparison with Fisher corpus systems. We additionally report frame-level classification accuracy (Acc) on a held out test set to compare DNNs as classifiers independent of the HMM decoder.

5.4. Number of hidden layers

We next compare performance of DNN systems while keeping total model size fixed and varying the number of hidden layers in the DNN. The optimal architecture for a neural network may change as the total number of model parameters changes. There is no a priori reason to believe that 5 hidden layers are optimal for all model sizes. Furthermore, there are no good general heuristics to select the number of hidden layers for a particular task. Table 5 shows DNN system performance for DNNs with 1, 3, 5, and 7 hidden layers for DNNs of at multiple total parameter counts.

The most striking distinction in terms of both frame classification and WER is the performance gain of deep models versus those with a single hidden layer. Single hidden layer models perform much worse than DNNs with 3 hidden layers or more. Among deep models there are much smaller gains as a function of depth. Models with 5 hidden layers show a clear gain over those with 3 hidden layers, but there is little to no gain from a 7 hidden layer model when compared with a 5 hidden layer model. These results suggest that for this task 5 hidden layers may be deep enough to achieve good performance, but that DNN depth taken further does not increase performance. It's also interesting to note that DNN depth has a much larger impact on performance than total DNN size. For this task, it is much more important to select an appropriate number of hidden layers than it is to choose an appropriate total model size.

For each total model size there is a slight decrease in frame classification in 7 layer DNNs as compared with 5 hidden layer DNNs. This trend of decreasing frame-level performance is also present in the training set, which suggests that as networks become very deep it is more difficult to minimize the training objective function. This is evidence for a potential confounding factor when building DNNs. In theory deeper DNNs should be able to model more complex functions than their shallower counterparts, but in practice we found that depth can act as a regularizer due to the difficulties in optimizing very deep models.

6. WER and frame classification error analysis

We now decompose our task performance metrics of frame classification accuracy and WER into their constituent components to gain a deeper understanding of how models compare to one another. This analysis attempts to uncover differences in models which achieve similar aggregate performance. For example, two systems which have the same final WER may have different rates of substitutions, deletions, and insertions – the constituent components of the WER metric.

Fig. 6 shows decomposed WER performance of HMM-DNN systems of varying DNN size. Each HMM-DNN system uses a DNN with 5 hidden layers, these are the same HMM-DNN systems reported in Table 5. We see that decreases

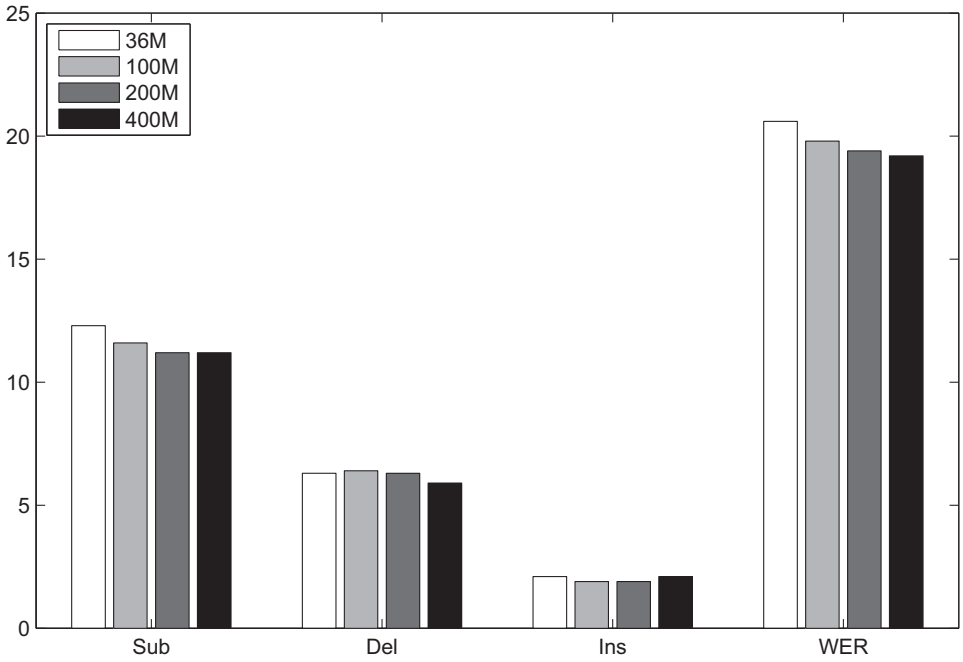

Fig. 6. Eval2000 WER of 5 hidden layer DNN systems of varying total parameter count. WER is broken into its sub-components – insertions, substitutions, and deletions.

in overall WER as a function of DNN model size are largely driven by lower substitution rates. Insertions and deletions remain relatively constant across systems, and are generally the smaller components of overall WER. Decreased substitution rates should be a fairly direct result of improving acoustic model quality as the system becomes more confident in matching audio features to senones. While the three WER sub-components are linked, it is possible that insertions and deletions are more of artifacts of other system shortcomings such as out of vocabulary words (OOVs) or a pronunciation dictionary which does not adequately capture pronunciation variations.

We next analyze performance in terms of frame-level classification grouped by phoneme. When understanding senone classification we can think of the possible senone labels as leaves from a set of trees. Each phoneme acts as the root of a different tree, and the leaves of a tree correspond to the senones associated with the tree's base phoneme.

Fig. 7 shows classification percentages of senones grouped by their base phoneme. The DNNs analyzed are the same 5 hidden layer models presented in our WER analysis of Fig. 6 and Table 5. The total height of each bar reflects its percentage of occurrence in our data. Each bar is then broken into three components – correct classifications, errors within the same base phoneme, and errors outside the base phoneme. Errors within the base phoneme correspond to the examples where the true label is a senone from a particular base phone, e.g. *ah*, but the network predicts an incorrect senone label also rooted in *ah*. The other type of error possible is predicting a senone from a different base phoneme. Together these three categories – correct, same base phone, and different base phone – additively combine to form the total set of senone examples for a given base phone.

The rate of correct classifications is non-decreasing as a function of DNN model size for each base phoneme. The overall increasing accuracy of larger DNNs comes from small correctness increases spread across many base phonemes. Across phonemes we see substantial differences in within-base-phoneme versus out-of-base-phoneme error rates. For example, the vowel *iy* has a higher rate of within-base-phoneme errors as compared to the fairly similar vowel *ih*. Similarly, the consonants *m*, *k*, and *d* have varying rates of within-base versus out-of-base errors despite having similar total rates of base phoneme occurrence in the data. We note that our DNNs generally exhibit similar error patterns to those observed with DNN acoustic models on smaller corpora (Huang et al., 2014). However, due to the challenging nature of our corpus we observe overall lower phone accuracies than those found in previous work. Performance as a function of model size appears to change gradually and fairly uniformly across phonemes, rather than larger models improving upon only specific phonemes, perhaps at the expense of performance on others.

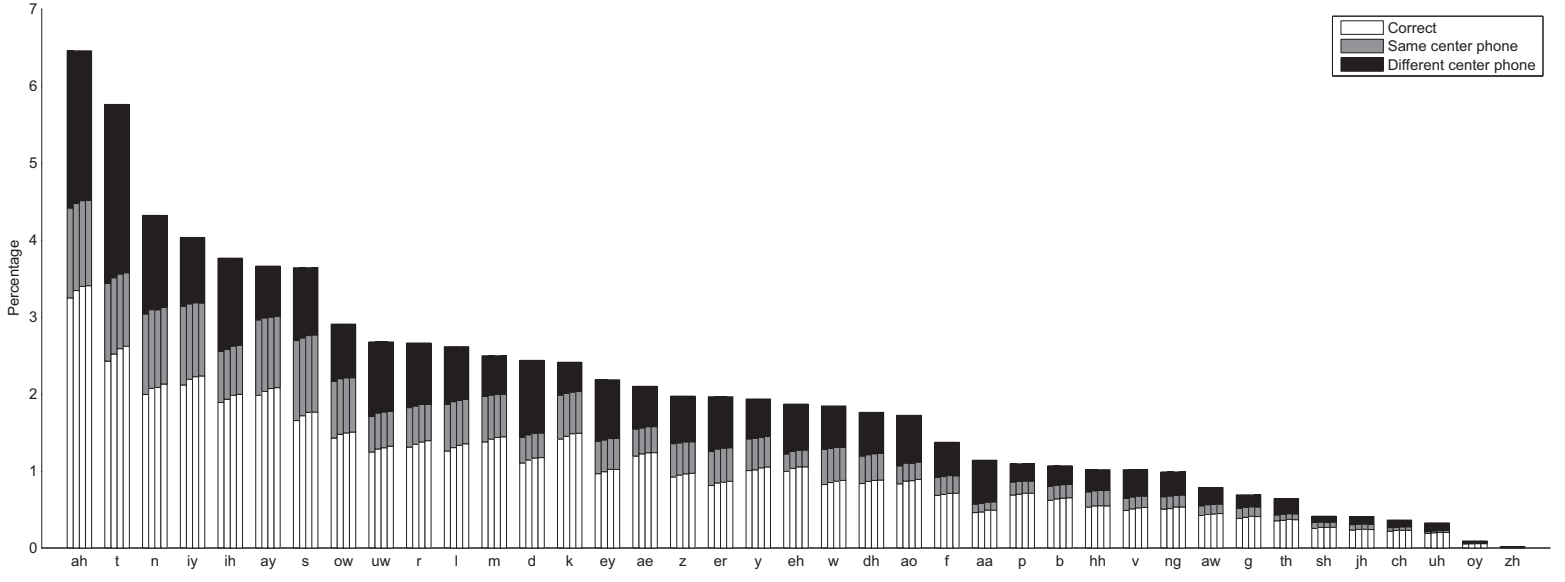

Fig. 7. Senone accuracy of 5 hidden layer DNN systems of varying total parameter count. Accuracy is grouped by base phone and we report the percentage correct, mis-classifications which chose a senone of the same base phone, and mis-classifications which chose a senone of a different base phone. The total size of the combined bar indicates the occurrence rate of the base phone in our data set. Each base phone has five bars, each representing the performance of a different five layer DNN. The bars show performance of DNNs of size 36M 100M 200M and 400M from left to right. We do not show the non-speech categories of silence, laughter, noise, or OOV which comprise over 20% of frames sampled.

7. Conclusion

The multi-step process of building neural network acoustic models comprises a large design space with a broad range of previous work. Our work sought to address which of the most fundamental DNN design decisions are most relevant for final ASR system performance. We found that increasing model size and depth are simple but effective ways to improve WER performance, but only up to a certain point. For the Switchboard corpus, we found that regularization can improve the performance of large DNNs which otherwise suffer from overfitting problems. However, a much larger gain was achieved by utilizing the combined 2100 hr training corpus as opposed to applying regularization with less training data.

Our experiments suggest that the DNN architecture is quite competitive with specialized architectures such as DCNNs and DLUNNs. The DNN architecture outperformed other architecture variants in both frame classification and final system WER. While previous work has used more specialized features with locally connected models, we note that DNNs enjoy the benefit of making no assumptions about input features having meaningful time or frequency properties. This enables us to build DNNs on whatever features we choose, rather than ensuring our features match the assumptions of our neural network. We found that DLUNNs performed slightly better and DCNNs, and may be an interesting approach for specialized acoustic modeling tasks. For example, locally untied models may work well for robust or reverberant recognition tasks where particular frequency ranges experience interference or distortion.

We trained DNN acoustic models with up to 400M parameters and 7 hidden layers, comprising some of the largest models evaluated to date for acoustic modeling. When trained with the simple NAG optimization procedure, these large DNNs achieved clear gains on both frame classification and WER when the training corpus was large. An analysis of performance and coding properties revealed a fairly gradual change in DNN properties as we move from smaller to larger models, rather than finding some phase transition where large models begin to encode information differently from smaller models. Overall, total network size, not depth, was the most critical factor we found in our experiments. Depth is certainly important with regard to having more than one hidden layer, but differences among DNNs with multiple hidden layers were fairly small with regard to all metrics we evaluated. At a certain point it appears that increasing DNN depth yields no performance gains, and may indeed start to harm performance. When applying DNN acoustic models to new tasks it appears sufficient to use a fixed optimization algorithm, we suggest NAG, and cross-validate over total network size using a DNN of at least three hidden layers, but no more than five. Based on our results, this procedure should instantiate a reasonably strong baseline system for further experiments, by modifying whatever components of the acoustic model building procedure researchers choose to explore.

References

Abdel-Hamid, O., Mohamed, A., Jang, H., Penn, G., 2012. Applying convolutional neural networks concepts to hybrid nn-hmm model for speech recognition. In: ICASSP.

Bourlard, H., Morgan, N., 1993. Connectionist Speech Recognition: A Hybrid Approach. Kluwer Academic Publishers, Norwell, MA.

Caruana, R., Lawrence, S., Giles, L., 2000. Overfitting in neural nets: backpropagation, conjugate gradient, and early stopping. In: NIPS.

Cieri, C., Miller, D., Walker, K., 2004. The fisher corpus: a resource for the next generations of speech-to-text. In: LREC, vol. 4, pp. 69–71.

Coates, A., Huval, B., Wang, T., Wu, D., Ng, A., Catanzaro, B., 2013. Deep learning with COTS HPC systems. In: ICML.

Dahl, G., Yu, D., Deng, L., Acero, A., 2012. Context-dependent pre-trained deep neural networks for large vocabulary speech recognition. IEEE Trans. Audio Speech Lang. Process. 20, 30–42.

Dahl, G., Sainath, T., Hinton, G., 2013. Improving deep neural networks for LVCSR using rectified linear units and dropout. In: ICASSP.

Graves, A., Jaitly, N., 2014. Towards end-to-end speech recognition with recurrent neural networks. In: ICML.

Graves, A., Mohamed, A., Hinton, G., 2013. Speech recognition with deep recurrent neural networks. In: ICASSP. IEEE, pp. 6645–6649.

Hermansky, H., Ellis, D., Sharma, S., 2000. Tandem connectionist feature extraction for conventional hmm systems. In: ICASSP, vol. 3. IEEE, pp. 1635–1638.

Hinton, G., Deng, L., Yu, D., Dahl, G., Mohamed, A., Jaitly, N., et al., 2012. Deep neural networks for acoustic modeling in speech recognition. IEEE Signal Process. Mag. 29, 82–97.

Huang, Y., Yu, D., Liu, C., Gong, Y., 2014. A comparative analytic study on the gaussian mixture and context dependent deep neural network hidden Markov models. In: INTERSPEECH.

Kingsbury, B., Sainath, T., Soltau, H., 2012. Scalable minimum Bayes risk training of deep neural network acoustic models using distributed hessian-free optimization. In: INTERSPEECH.

Krizhevsky, A., Sutskever, I., Hinton, G., 2012. ImageNet classification with deep convolutional neural networks. In: NIPS.

LeCun, Y., Bottou, L., Bengio, Y., Haffner, P., 1998. Gradient-based learning applied to document recognition. Proc. IEEE 86 (11), 2278–2324.

Lee, H., Grosse, R., Ranganath, R., Ng, A.Y., 2009. Convolutional deep belief networks for scalable unsupervised learning of hierarchical representations. In: ICML. ACM, pp. 609–616.

Li, X., Wu, X., 2015. Constructing long short-term memory based deep recurrent neural networks for large vocabulary speech recognition.
Liao, H., McDermott, E., Senior, A., 2013. Large scale deep neural network acoustic modeling with semi-supervised training data for YouTube video transcription. In: ASRU.
Maas, A., Hannun, A., Ng, A., 2013. Rectifier nonlinearities improve neural network acoustic models. In: ICML Workshop on Deep Learning for Audio, Speech, and Language Processing.
Maas, A.L., Xie, Z., Jurafsky, D., Ng, A.Y., 2015. Lexicon-free conversational speech recognition with neural networks. In: HLT-NAACL.
Nesterov, Y., 1983. A method of solving a convex programming problem with convergence rate O (1/k2). In: Soviet Mathematics Doklady, vol. 27, pp. 372–376.
Plaut, D., 1986. Experiments on learning by back propagation.
Povey, D., Ghoshal, A., Boulianne, G., Burget, L., Glembek, O., Veselý, K., et al., 2011. The Kaldi speech recognition toolkit. In: ASRU.
Renals, S., Morgan, N., Bourlard, H., Cohen, M., Franco, H., 1994. Connectionist probability estimators in hmm speech recognition. IEEE Trans. Speech Audio Process. 2 (1), 161–174.
Rumelhart, D.E., Hinton, G.E., Williams, R.J., 1986. Learning internal representations by error propagation. In: Parallel Distributed Processing: Explorations in the Microstructures of Cognition, Volume 2: Psychological and Biological Models, vol. 76, p. 1555.
Sainath, T., Kingsbury, B., Sindhwani, V., Arisoy, E., Ramabhadran, B., 2013. Low-rank matrix factorization for deep neural network training with high-dimensional output targets. In: ICASSP.
Sainath, T., Kingsbury, B., Mohamed, A., Dahl, G., Saon, G., Soltau, H., et al., 2013. Improvements to deep convolutional neural networks for LVCSR. In: ASRU.
Sainath, T.N., Kingsbury, B., Saon, G., Soltau, H., Mohamed, A., Dahl, G., et al., 2014. Deep convolutional neural networks for large-scale speech tasks. Neural Netw. 64, 39–48.
Sak, H., Senior, A., Beaufays, F., 2014. Long short-term memory recurrent neural network architectures for large scale acoustic modeling. In: INTERSPEECH.
Saxe, A., McClelland, J., Ganguli, S., 2013. Learning hierarchical category structure in deep networks. In: CogSci.
Sutskever, I., 2013. Training recurrent neural networks (Ph.D. thesis). University of Toronto.
Sutskever, I., Martens, J., Dahl, G., Hinton, G., 2013. On the importance of momentum and initialization in deep learning. In: ICML.
Vesel, K., Ghoshal, A., Burget, L., Povey, D., 2013. Sequence-discriminative training of deep neural networks. In: INTERSPEECH. pp. 2345–2349.
Vinyals, O., Ravuri, S.V., Povey, D., 2012. Revisiting recurrent neural networks for robust ASR. In: ICASSP. IEEE, pp. 4085–4088.
Wager, S., Wang, S., Liang, P., 2013. Dropout training as adaptive regularization. In: NIPS.
Waibel, A., Hanazawa, T., Hinton, G., Shikano, K., Lang, K.J., 1989. Phoneme recognition using time-delay neural networks. IEEE Trans. Acoust. 37 (3), 328–339.
Weng, C., Yu, D., Watanabe, S., Juang, B., 2014. Recurrent deep neural networks for robust speech recognition. In: ICASSP.
Wiesler, S., Golik, P., Schlüter, R., Ney, H., 2015. Investigations on sequence training of neural networks.
Zeiler, M., Ranzato, M., Monga, R., Mao, M., Yang, K., Le, Q., et al., 2013. On rectified linear units for speech processing. In: ICASSP.