FINE CONTEXT, LOW-RANK, SOFTPLUS DEEP NEURAL NETWORKS FOR MOBILE SPEECH RECOGNITION

Andrew Senior *Xin Lei*

Google Inc., USA

{andrewsenior,xinlei}@google.com

ABSTRACT

We investigate the use of large state inventories and the softplus nonlinearity for on-device neural network based mobile speech recognition. Large state inventories are achieved by less aggressive context-dependent state tying, and made possible by using a bottleneck layer to contain the number of parameters. We investigate alternative approaches to the bottleneck layer, demonstrate the superiority of the softplus nonlinearity and investigate alternatives for the final stages of the training algorithm. Overall we reduce the word error rate of the system by 9% relative. The techniques are also shown to work well for large acoustic models for cloud-based speech recognition.

Index Terms— Deep neural networks, hybrid neural network speech recognition, Voice Search, mobile speech recognition, embedded recognizer, low-rank approximation, singular value decomposition, softplus nonlinearity.

1. INTRODUCTION

In recent years speech recognition has become an important modality for interacting with mobile devices. Speech is used for general text entry but is particularly useful for interacting with certain applications, such as search, maps search or translation. While the majority of mobile speech recognition systems operate "in the cloud" by sending the speech signal over a network to a server which returns a transcription, there are several scenarios where local speech recognition is useful even when there is no network connection, for instance contact dialing, dictating text messages or offline documents and command-and-control of the device. For this reason we are interested in building a high accuracy real-time speech recognizer that can run on a mobile device. Previously we have described [1] an embedded speech recognizer that uses deep neural networks (DNNs) to transcribe speech on a mobile phone.

In this paper we describe an investigation into improving the performance of this DNN-based speech recognizer by improving the acoustic model. In particular, in Section 3, we focus on increasing the state inventory of the model and compensating for the increase in the number of parameters by using a low-rank approximation to the weight matrices. We have shown that this technique is effective in large-scale acoustic models for transcribing YouTube audio [2]. In Section 3.3 we demonstrate that a linear bottleneck layer outperforms an equivalent nonlinear bottleneck layer, and in Section 3.2 discuss practical implementation of quantized versions of these networks.

In Section 4 we apply the softplus nonlinearity to speech recognition for the first time and find that it gives better Word Error Rates (WERs) and faster convergence than Rectified Linear Units (ReLU). Finally Section 5 explores variations of fine tuning to see if the final stages of training can be improved.

2. BACKGROUND

In a previous paper [1] we described how a Gaussian mixture model (GMM) based speech recognizer designed to run on a mobile phone was replaced with a DNN-based recognizer that gave a 27.5% lower WER but still ran in real-time on the target device and required less memory. The deep neural network model used in that work had 2.7M parameters, consisting of six sigmoid hidden layers with 512 hidden units each and softmax outputs for the 2000 context-dependent state posteriors. The network processed a context window of 16 (10 past and 5 future) frames of speech, each represented with 40 dimensional log mel filterbank energies taken from 25ms windows every 10ms. In this work we start with the same network configuration but modify it by changing the nonlinearity and the output layer. Following the success of ReLUs [3] for speech, we build our baseline models with this nonlinearity.

The systems in the current and previous paper are trained on a US English data set of 3M anonymized utterances (1,750 hours or about 1 billion frames) collected from live voice search and dictation traffic. The utterances are hand-transcribed and force-aligned with a 70M parameter model with 14,000 context-dependent (CD) states. Training is done on a GPU using stochastic gradient descent with 200-frame minibatches. WERs are evaluated on a test set containing 23,000 similar utterances. The language model used is a 23 million n-gram model trained with up to 5-grams, and a vocabulary of 2.6 million words.

978-1-4799-2893-4/14/$31.00 ©2014 IEEE

Output States	Word error rate	# of Parameters
2000	15.1	$2.7M$
4000	14.2	$3.7M$
8000	13.5	$5.7M$
14000	13.4	$9.0M$

Table 1: Word error rates (for ReLU networks) decrease as the number of output states is increased, but the number of parameters increases.

For this work we use the number of parameters as a proxy for the final speed of the system, though we do not need to compute all of the final outputs for every frame, since some are not needed in the search because of beam constraints.

3. INCREASED STATE INVENTORY

The previously described model used 2000 output states as a trade-off between the increased accuracy of a large inventory and the slower inference speed caused by a larger number of parameters. The 2000 state inventory was created by conventional decision-tree based clustering of triphones in a GMM system. The decision trees were pruned at different levels to generate a nested series of state inventories with 14,000, 8,000, 4,000, 2,000 and 1,000 states, with a simple many:one mapping from the labels of the 14000 set to the smaller inventories.

As shown in Table 1, increasing the state inventory leads to a decrease in word error rates, but with a corresponding increase in the number of parameters.

3.1. Bottleneck layer

Sainath *et al.* [4] showed that, just as a matrix can be approximated as a product of two matrices with lower rank, a large output layer can be approximated by a linear layer (a layer with no nonlinearity) followed by a softmax layer, the product of whose weight matrices approximates the original weight matrix. If the number of linear units is small enough (termed a bottleneck), this low-rank substitution can bring down the total number of parameters, at the cost of limiting the modelling power of the layer. However, if the weights of the different outputs are highly correlated, as these authors suggest they are, since many of them are modelling slightly different context-dependent variations of the same context-independent state, then the loss in modelling power may have no impact on, or even (by regularization) improve WER. Reducing the number of parameters reduces the storage size of the networks and their computation time in training and inference. The technique can be applied to other layers, but there is less advantage in layers where the numbers of inputs and outputs are similar, and where there is less correlation in the weights. Sainath *et al.* [4] showed no advantage from applying the technique in other layers.

Output States	Low-Rank approximation				Full Rank
	128	192	256	320	
2000					15.1 $2.7M$
4000	14.6 $2.2M$	14.3 $2.5M$	14.2 $2.8M$	14.2 $3.1M$	14.2 $3.7M$
8000	14.1 $2.7M$	13.8 $3.3M$	13.7 $3.8M$	13.7 $4.4M$	13.5 $5.7M$
14000	13.7 $3.5M$	13.6 $4.5M$	13.5 $5.4M$	13.5 $6.4M$	13.4 $9.0M$

Table 2: Word error rates (upper) for ReLU networks with linear bottleneck layers decrease as the size of the linear layer is increased, increasing the number of parameters (lower).

Following this work, we consider a low-rank approximation to the final layer to reduce the number of parameters. We train models with an additional hidden layer with fewer outputs than the other hidden layers and with no nonlinearity. This layer's outputs are input to the final softmax layer. We initialize these layers with the same low variance random initialization as ReLU, and use the same low learning rate (0.005) as has been used for ReLU, with exponential decay (by a factor of 10 every 6 billion frames [5]). Word error rates are shown in Table 2, and are plotted against number of parameters in Figure 1. The baseline model has 2000 output states, with a WER of 15.1% and 2.67M parameters. The results for the 8000 output, rank 128 model (2.74M parameters) show that we can achieve a 1% absolute word error rate improvement over the baseline by a modest (2.7%) increase in the number of parameters.

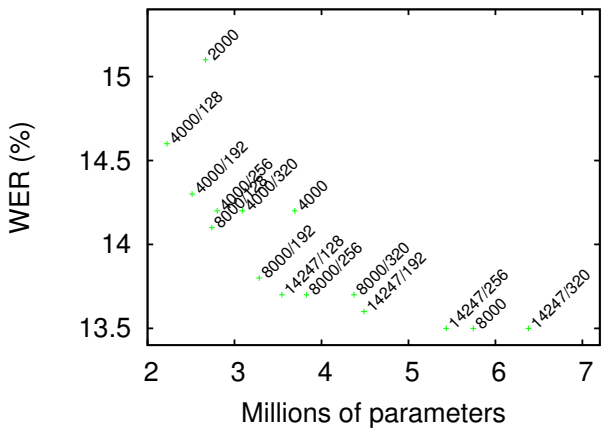

Fig. 1: Graph of Word Error Rate against number of parameters for the networks of Table 2 labeled with number of outputs / bottleneck size.

3.2. Quantized linear networks

As previously described [6], evaluating networks in real time is computationally challenging and we use a number of ways to maximize the size of network that we can evaluate in a given computation budget. One of the most significant of these is 8-bit quantization of weights and activations which allows for 16-way-parallel execution of multiplications. Switching to ReLUs presents a challenge for the quantization of the activations, because the ReLU outputs are not bounded above. However, we find that activations rarely if ever exceed 16, so we can quantize activations such that the ReLU saturates above 16. Since the ReLU layers are linear when non-zero, we can in fact scale the first layer's weights and biases down by a factor of 16 so that activations of all subsequent layers are scaled down by a factor of 16. The first non-ReLU layer (in our case the softmax) has its weights (but not biases) scaled up by a factor 16 to compensate.

The addition of a linear layer complicates things further, since the activations are not bounded above or below. However, we find in practice that activations fall within $[-8, 8]$ so by adding 8 to all the linear layer's biases, and compensating by changing the biases of the softmax layer, we can operate with always-positive activations, and in fact this quantized shifted linear layer can actually be implemented as a ReLU layer.

3.3. Nonlinear, low-rank bottlenecks

The success of deep learning, the addition of a linear low-rank bottleneck layer, and our ReLU replacement in the quantized case, raise the question of whether we could achieve greater modelling power by using a nonlinearity in the additional layer. We trained more networks from random initialization with a ReLU nonlinearity on the bottleneck layer units. The results in Table 3 show that the linear bottleneck layer tends to perform better than the ReLU bottleneck.

3.4. Low-rank decomposition of trained networks

Given an already-trained softmax layer, we can make a low-rank factorization of its weight matrix W, and replace it with a linear layer with weights W_1 followed by a softmax layer with weights W_2 where $W_1 * W_2 \approx W$. A common way to compute the factorization is to compute the singular value decomposition (SVD) or W where $W = USV^T$ with S being a diagonal matrix of k singular values. By choosing to truncate the number of preserved singular values to the desired rank R (and trimming U and V to have R columns) then we have a low-rank approximation to W, in particular we choose $W_1 = U_R\sqrt{S_R}$ and $W_2 = \sqrt{S_R}V_R^T$. Xue, Li and Gong [7] have shown that performing such an approximation hurts the accuracy of a neural network temporarily, but further fine-tuning by continuing training of the new network can recover some of the lost accuracy and, in their experiments, outperform the bottleneck network of the same shape and size trained from random initialization.

We have applied this technique on the output layers with a variety of bottleneck ranks, as shown in Table 4. We find that for our task, this technique always leads to an equal or higher error rate than the method of Section 3.1. Since it also takes longer to train (both in number of epochs and computation required for the earlier epochs) and adds an additional hyperparameter (when to make the low-rank approximation) compared to the technique of Sainath *et al.* [4], we prefer the latter.

4. THE SOFTPLUS NONLINEARITY

Having validated the approaches using ReLU units, we carried out the same experiments using the softplus nonlinearity

$$f(x) = log(1 + exp(x)) \quad (1)$$

described by Glorot *et al.* [8], which can be considered as a smooth approximation to the ReLU function that is always differentiable and has a non-zero gradient below 0. Previous work [9] showed no gain from using a different non-saturating approximation to the ReLU which they termed the *leaky ReLU*: $f(x) = x(x > 0)$; $f(x) = 0.01x(x \leq 0)$. The main reason for using the softplus is that it propagates gradients throughout its domain. This can make networks easier to train since ReLUs propagate no gradient in the saturated $x < 0$ domain, whence they are not able to recover.

Table 4 shows that the softplus nonlinearity always gives a WER equal to or lower than that when using ReLU, and again we find that training with a bottleneck from initialization always performs at least as well as when making a low-rank approximation during training. We also find again (not shown) that putting a nonlinearity in the bottleneck layer always gives the same or higher WER. Further we find that softplus converges faster than ReLU, reaching the lowest WER after about 12B frames of training, compared to about 16B for the ReLU, presumably because the ReLU's gradients are frequently zero, making training less effective.

The softplus nonlinearity and large state inventory with a linear bottleneck were also found to be effective for a large acoustic model as shown in Table 5.

Nonlinearity	Bottleneck	Outputs	Params.	WER
Rectified Linear	None	14,000	85.0	10.1
Rectified Linear	1024	33,000	84.8	10.0
Softplus	1024	33,000	84.8	**9.8**

Table 5: WERs for three large (85M parameter) networks on the same test set as above. These three networks have 8 layers of 2560 hidden units and a context window of 20 past and 5 future stacked frames.

Output States	Low-Rank approximation (Linear/ReLU)				Full Rank
	128	192	256	320	
2000					15.1
4000	**14.6** / 14.7	**14.3** / 14.5	**14.2** / 14.3	14.2 /	14.2
8000	14.1 / 14.1	**13.8** / 14.0	**13.7** / 13.8	**13.7** / 14.0	13.5
14000	**13.7** / 13.9	13.6 / 13.6	13.5 / 13.5	13.5 /**13.4**	13.4

Table 3: Making the linear bottleneck layer (left) a ReLU layer (right) hurts performance for the lower dimension bottlenecks.

Outputs	Nonlinearity	Bottleneck Creation	Bottleneck size				Full Rank
			128	192	256	320	
8k	ReLU	From-scratch	14.1	13.8	13.7	13.7	13.5
8k	Sigmoid	From-scratch	14.1	13.8	13.6	13.6	13.6
8k	Softplus	From-scratch	**13.8**	**13.6**	**13.5**	**13.5**	13.5
8k	ReLU	14B frames	14.2	14.0	13.8		
8k	Softplus	7B frames	**13.8**				
14k	ReLU	From-scratch	13.7	13.6	13.5	13.4	13.4
14k	Softplus	From-scratch	**13.6**	**13.4**	**13.4**	**13.3**	**13.2**

Table 4: WERs for 8k outputs when making a low-rank approximation (retaining a variable number of singular values) part way through training at full rank compared to training the same sized bottleneck networks from scratch. The table shows both the initial ReLU nonlinearity and the softplus nonlinearity of Section 4 for networks with 8k and 14k outputs.

Techniques	WER	Frame accuracy
Baseline	**13.8**	49.5
Squared Error	14.1	49.7
Thresholding	14.2	49.0

Table 6: Percentage WERs and frame accuracies (on a 200,000 frame held-out set) when applying various fine-tuning strategies to the baseline softplus 128-rank bottleneck, 8k output network.

5. TRAINING REFINEMENTS

In the past it has been reported [1, 10] that thresholding small weights to zero during training improves performance, as a simple regularizer. We apply the same technique here to the best softplus 8k, 128-rank network, choosing the threshold (0.04) so that about 30% of the weights in most layers are set to zero with a higher fraction of the input layer. The thresholding is applied to every weight at every update after 10B frames. We also follow Golik *et al.* [11] who demonstrated better WERs when training softmax outputs to a squared error criterion after initial fine tuning with the standard cross-entropy criterion.

Table 6 shows that neither of these refinements helps: training towards the squared error criterion increases the frame accuracy (0.2%), but also increases the WER by 0.3%. For this model, thresholding the weights also hurts the WER, and we suggest that this is because the low-rank layer has less redundancy where even the small weights are critical.

6. CONCLUSIONS

We have shown that increasing the state inventory consistently decreases the WER on our task. By using a bottleneck layer we were able to keep the number of parameters almost the same (2.6% increase) and still achieve a 1% absolute reduction in WER. We have demonstrated that the softplus nonlinearity is superior to ReLU, bringing a further reduction in word error rate and faster training. These three techniques together were able to bring the Word Error Rate for a 2.7M parameter network down to 13.8%, a 9% reduction relative to the initial network. Softplus has outperformed ReLU, both for networks for mobile devices and for much larger networks for cloud-based servers. We expect softplus to receive wide application for speech recognition.

We found that using a nonlinearity in the bottleneck layer increased the WER, and found no advantage to constructing a low-rank approximation during training, relative to training with the corresponding bottleneck from the start. Using a squared error objective increased the WER, as did the weight thresholding which helped in our previous work.

7. REFERENCES

[1] X. Lei, A. Senior, A. Gruenstein, and J. Sorenson, "Accurate and compact large vocabulary speech recognition on mobile devices," in *Proc. Interspeech*, 2013.

[2] H. Liao, E. McDermott, and A. Senior, "Large scale deep neural network acoustic modeling with semi-

supervised training data for YouTube video transcription," in *Proc. ASRU*, 2013.

[3] M.D. Zeiler, M. Ranzato, R. Monga, M. Mao, K. Yang, Q.V. Le, P. Nguyen, A. Senior, V. Vanhoucke, J. Dean, and G.E. Hinton, "On rectified linear units for speech processing," in *ICASSP*, 2013.

[4] T.N. Sainath, B. Kingsbury, V. Sindhwani, E. Arisoy, and B. Ramabhadran, "Low-rank matrix factorization for deep neural network training with high-dimensional output targets," in *Proc. ICASSP*, 2013.

[5] A. Senior, G. Heigold, M. Ranzato, and K. Yang, "An empirical study of learning rates in deep neural networks for speech recognition," in *Proc. ICASSP*, 2013.

[6] V. Vanhoucke, A. Senior, and M. Z. Mao, "Improving the speed of neural networks on CPUs," in *Proc. Deep Learning and Unsupervised Feature Learning NIPS Workshop*, 2011.

[7] J. Xue, J. Li, and Y. Gong, "Restructuring of deep neural network acoustic models with singular value decomposition," in *Proc. Interspeech*, 2013.

[8] X. Glorot, A. Bordes, and Y. Bengio, "Deep sparse rectifier neural networks," in *Proc. AISTATS*, 2011.

[9] A. Maas, A. Jannun, and A. Ng, "Rectifier nonlinearities improve neural network acoustic models," in *ICML Workshop on Deep Learning for Audio, Speech and Language Processing*, 2013.

[10] D. Yu, F. Seide, G. Li, and L. Deng, "Exploiting sparseness in deep neural networks for large vocabulary speech recognition," in *ICASSP*, 2012.

[11] P. Golik, P. Doetsch, and H. Ney, "Cross-entropy vs. squared error training: a theoretical and experimental comparison," in *Proc. Interspeech*, 2013.