Model compression applied to small-footprint keyword spotting

George Tucker[1], Minhua Wu[1,2], Ming Sun[1], Gengshen Fu[1], Sankaran Panchapagesan[1], Shiv Vitaladevuni[1]

tuckeg@amazon.com, shivnaga@amazon.com [1]Amazon, [2]Johns Hopkins University (work done while at Amazon)

Introduction

Recently, a number of devices and services have enabled fully voice-based interfaces, such as the Google Now, iPhone 6s, and the Amazon Echo. For privacy reasons, these devices rely on the user to preface their commands with a keyword, such as "Alexa". Accurate on-device keyword spotting is critical to usability. In this work, we focused on keyword spotting systems (KWS) for small-footprint devices. In particular, we investigated the use of low rank weight matrices [1,2] and knowledge distillation [3, 4] applied to the deep neural network (DNN) based KWS system described in [5]. We found that these techniques combine to give significant reductions in false alarms (FAs) and misses (**~10% reduction in FAs at a fixed miss rate**).

Baseline KWS system

For our experiments, we used the KWS system described in [5]. Briefly, the DNN classifies each frame as keyword or background, then we average the posteriors over 30 frames with a sliding window, and use the maximum smoothed value over a 100 frame sliding window.

The baseline DNN was composed of 4 hidden layers with 248 nodes and sigmoid activations. Preliminary experiments with ReLu activations did not show significant differences for this work. The input to the DNN was stacked 20 dimensional LFBE features extracted from 25 ms frames (shifted by 10 ms). We stacked 20 left context and 10 right context frames. We used an asymmetric context because right context increases latency.

We trained DNNs with distributed asynchronous SGD [6]. We used a performance based learning rate, where the learning rate is halved every time performance degrades on the development set. The initial learning rate and batch size were tuned on the development set.

Methods

Low-rank weight matrices

Low-rank weight matrices, implemented as linear bottlenecks (BN), have been used in DNNs for mobile speech recognition [1], where they reduce the number of multiplications, at the expense of representational power. Previous work has shown that reducing the rank can have a regularizing effect [2]. However, most previous work has restricted the bottleneck to the input and output weight matrices, where the majority of the parameters are. In our system, the input and hidden layers are similar in size while the output layer is small.

We explored the tradeoff between the number of nodes per hidden layer and the size of the linear bottleneck. To train the reduced rank DNN, we:

1. Train a full-rank DNN.
2. Starting from the weight matrix connected to the input layer, one layer at a time, add a linear bottleneck initialized by the SVD of the full-rank matrix. Then, train for one epoch, and repeat this step until all matrices are decomposed.
3. Train for an additional 20 epochs.

We found that initializing the linear bottleneck with the SVD was essential for convergence.

Knowledge distillation

Recent work has shown that a computationally efficient model can be trained to mimic a teacher ensemble of larger models [3, 4]. This is referred to as knowledge distillation (KD). We applied it to train the KWS DNN. Specifically, we trained 8 600 nodes per hidden layer DNNs with different random initializations and formed an ensemble model by averaging posteriors. The KD DNN was trained with the weighted criterion:

$$\lambda \sum_i \log(p_i) * t_i + \frac{(1-\lambda)}{T^2} \sum_i \log(p_i(T)) * q_i(T)$$

$$p_i(T) = p_i^{1/T} / \sum_j p_j^{1/T}, q_i(T) = q_i^{1/T} / \sum_j q_j^{1/T}$$

where p_i is posterior output for class i, t_i is a binary indicator of the correct label, q_i is the posterior output from the teacher model, and λ and T are hyperparameters.

Results

For training and evaluation, we used an in-house corpus with many instances of the keyword "Alexa" and background speech. For business reasons, we cannot divulge specifics, but the observed differences on the test set are statistically significant. We retained 90% of the data for training, 1% for development, and 9% for testing.

Low-rank weight matrices

We trained models with 248, 300, 400, and 500 nodes per hidden layer while varying the size of the linear bottleneck layer. We found that the linear bottleneck improved performance, especially as the size increased (Fig. 1). Surprisingly, frame error rate continued to decrease as the number of parameters exceeded the number of parameters in the model without linear bottlenecks. Future work will investigate this. Importantly, because the bottleneck is linear, we can always combine the linear transforms, so the test time network will never have more parameters than the network without linear bottlenecks.

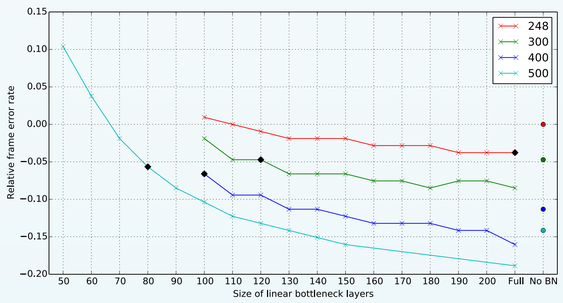

Figure 1: Relative frame error rate on the test set vs. size of linear bottleneck layers. Full corresponds to the models with bottleneck layers of equal size to the hidden layers. No BN corresponds to the models without linear bottlenecks. The black markers correspond to models that have comparable multiplications to the baseline 248 nodes per hidden layer model.

Without exceeding the baseline model multiplication budget, the 400 node per hidden layer with 100 node bottlenecks (denoted 400 x 100 in Fig. 2) is the best configuration (6.6% relative reduction in frame error rate, Fig. 1)

T \ λ	0.8	0.6	0.4	0.2	0
1	-0.019	-0.009	-0.028	**-0.047**	-0.038
2	0.047	-0.019	-0.028	-0.019	-0.028

Table 1: Relative frame error rate (compared to the baseline) on the test set as KD hyperparameters are varied.

Knowledge distillation

We trained a 248 node per hidden layer DNN using the KD criterion and observed comparable gains in frame error rate (Table 1).

Combined low-rank and knowledge distillation

Combining the two approaches provided further gains in frame error rate (Table 2).

T \ λ	0.8	0.6	0.4	0.2	0
1	-0.104	**-0.113**	**-0.113**	**-0.113**	-0.094
2	-0.104	-0.094	-0.094	-0.085	-0.075

Table 2: Relative frame error rate (compared to the baseline) on the test set as KD hyperparameters are varied combined with the best performing low-rank model (400 nodes per hidden layer, linear bottleneck of size 100).

Although both techniques reduced the DNN frame error, the KWS criterion is a related, but different sequence level objective function. We evaluated the top performing models on the KWS task (Fig. 2).

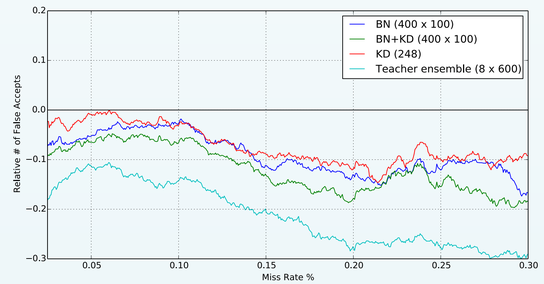

Figure 2: Relative DET curve on the KWS task (relative to the baseline model with 248 nodes per hidden layer).

We found that both techniques individually improve end-to-end KWS performance, and the combination further improves performance (~10% reduction in FAs at a fixed miss rate) all without increasing the CPU budget.

Literature Cited

[1] Sainath et al., "Low-rank matrix factorization for deep neural network training with high-dimensional output targets," ICASSP 2013.
[2] Wiesler et al., "Mean-normalized stochastic gradient for large-scale deep learning," ICASSP 2014.
[3] Hinton et al., "Distilling the Knowledge in a Neural Network," NIPS Deep Learning Workshop 2014.
[4] Ba et al., "Do Deep Nets Really Need to be Deep?," NIPS 2014.
[5] Chen et al., "Small-footprint keyword spotting using deep neural networks," ICASSP 2014.
[6] Strom, N., "Scalable Distributed DNN Training Using Commodity GPU Cloud Computing," INTERSPEECH 2015.