UNIVERSAL ACOUSTIC MODELING USING NEURAL MIXTURE MODELS

Amit Das[*], *Jinyu Li, Changliang Liu, Yifan Gong*

Microsoft, One Microsoft Way, Redmond, WA 98052

amitdas@illinois.edu, {jinyli, chanliu, ygong}@microsoft.com

ABSTRACT

Acoustic models are domain dependent and do not perform well if there is a mismatch between training and test conditions. As an alternative, the Mixture of Experts (MoE) model was introduced for multi-domain modeling. It combines the outputs of several domain specific models (or experts) using a gating network. However, one drawback is that the gating network directly uses raw features and is unaware of the state of the experts. In this work, we propose several alternatives to improve the MoE model. First, to make our MoE model state-aware, we use outputs of experts as inputs to the gating network. Then we show that vector based interpolation of the mixture weights is more effective than scalar interpolation. Second, we show that directly learning the mixture weights without using any complex gating is still effective. Finally, we introduce a hybrid attention model that uses the logits and mixture weights from the previous time step to generate the mixture weights at the current time. Our best proposed model outperforms a baseline model using LSTM based gating achieving about 20.48% relative reduction in word error rate (WER). Moreover, it beats an oracle model which picks the best expert for a given test condition.

Index Terms— mixture of experts, universal acoustic model, adaptation, attention, speech recognition

1. INTRODUCTION

Recent advances in speech recognition [1, 2] have been mostly due to the advent of deep learning algorithms such as Deep Neural Networks (DNN) [3], Convolutional Neural Networks (CNNs) [4], and Recurrent Neural Networks (RNN) [5, 6]. However, one fundamental limitation for neural networks to work well is that the joint distribution of the acoustics and the labels in the training data must match the distribution of the test data. If not, they tend to perform poorly [7]. Thus, acoustic models (AMs) tend to be highly domain dependent and domain specific or expert AMs have to be built individually for each domain. This hinders real-world deployment of AMs due to the presence of a wide variety of domains.

One way to alleviate this problem is to train a global model with data from K possible domains. A drawback of the global model is that it does not perform at par with an expert model. Moreover, the number of parameters and the training times tend to increase by several orders of magnitude. Yet another problem is that when additional amounts of data become available, the global model has to be retrained from scratch. A better solution is to combine the outputs of a mixture of pre-trained experts using a small gating network (switch) thereby reducing computational times and complexity. This is the MoE model proposed by Jacobs *et al*. [8] and is the first one in this direction. Their gating network mapped the input data to a probability distribution over the experts better known as mixture weights. The mixture weights determine the degree of relevance between the expert outputs and the input data. The expert outputs are linearly combined using the mixture weights to produce a unified output.

Following the work in [8], [9] used a mixture of recurrent experts. Later [10] suggested a tree-like hierarchical MoE model where the experts represented the leaves of the tree and the gates were stacked in a layer-wise fashion going up the tree. As an alternative, the authors in [11] proposed a deep MoE model where each layer constituted an MoE model (multiple experts and a single gate) and there were multiple such layers stacked one on top of the other. The MoE models have been used for noise-robust ASR [12–14], speech enhancement [15], and intent classification and slot tagging [16]. In [17, 18], HMM based MoE models were used for speaker adaptation where cluster weights were estimated to maximize the likelihood or minimize the minimum phone error. Similar idea has been applied to deep learning models as cluster adaptive training [19, 20]. More recently, Irie *et al*. [21] proposed the Recurrent Adaptive Mixture Model (RADMM) for training a language model in diverse domains. They constructed the gating network as a mixer long short-term memory (LSTM) [5] network. However, one problem with this approach is that the mixer LSTM is unaware of the state of the experts as it uses raw features to produce the mixture weights.

To address this drawback, we propose improving this MoE by using expert outputs (or their linear projections), instead of raw features, as inputs to the mixer LSTM. Furthermore, we show that vector based interpolation of expert outputs is more effective than scalar interpolation. Then, we experiment with directly learning the weights of the experts without using any mixer LSTM. Finally, we introduce a hybrid attention model that uses the logits and the mixture weights from the previous time step to generate the mixture weights at the current time. With all these components in place, our final model improves the baseline RADMM achieving a relative reduction in WER of about 20.48% across 8 different test conditions on Microsoft products. Moreover, it was also able to beat an oracle model which picks the best expert for a given test condition.

2. UNIVERSAL ACOUSTIC MODEL (UAM)

Since we are interested in modeling AMs across a wide variety of conditions, we will refer to these as UAMs. The training strategy of the UAMs is as follows. First, experts in each of the K domains are trained individually using large amounts of domain specific data. Then the UAM is constructed by combining the outputs of an ensemble of K experts with a gating network. While training the UAM, the parameters of the gating network are updated whereas the model parameters of the experts remain unchanged.

2.1. RADMM With Raw Features(RADMM:R)

This is our baseline model [21]. The RADMM network consists of the following components - multiple stacked LSTMs each representing a domain expert, a mixer LSTM, and a softmax output layer. We provide a brief outline of the forward pass operations. Assume there are K domains. An input feature vector $\mathbf{x}_t$ at time t is fed to K experts

[*]Work performed during an internship at Microsoft.

operating in parallel. The k^{th} expert is identified by LSTM_k where $k = 1, \cdots, K$. Feedforwarding $\mathbf{x}_t$ through LSTM_k results in,

$$(\mathbf{h}_t^{(k)}, \mathbf{c}_t^{(k)}) = \text{LSTM}_k(\mathbf{x}_t, \mathbf{h}_{t-1}^{(k)}, \mathbf{c}_{t-1}^{(k)}), \tag{1}$$

where $\mathbf{h}_t^{(k)}$ and $\mathbf{c}_t^{(k)}$ are the hidden output and the cell state respectively of the k^{th} expert LSTM_k at time t. The same input $\mathbf{x}_t$ is fed to a mixer LSTM (LSTM_{mix}) with projection layer followed by softmax activation. This generates the mixture (or expert) weight vector $\boldsymbol{\alpha}_t$

$$(\mathbf{h}_t^{(mix)}, \mathbf{c}_t^{(\text{mix})}) = \text{LSTM}_{\text{mix}}(\mathbf{x}_t, \mathbf{h}_{t-1}^{(\text{mix})}, \mathbf{c}_{t-1}^{(\text{mix})}), \tag{2}$$

$$\boldsymbol{\alpha}_t = \text{softmax}(\mathbf{W}_{\text{mix}}\mathbf{h}_t^{(\text{mix})} + \mathbf{b}_{\text{mix}}). \tag{3}$$

The mixture weight vector $\boldsymbol{\alpha}_t = [\alpha_t^{(1)} \cdots \alpha_t^{(K)}]$, is a vector of probabilities and hence $\sum_{k=1}^{K} \alpha_t^{(k)} = 1$. The mixture weight $\alpha_t^{(k)}$ determines the importance of the k^{th} expert in producing the unified output $\mathbf{s}_t$. Thus, the mixture weights are used to linearly combine the hidden outputs from the experts using,

$$\mathbf{s}_t = \sum_{k=1}^{K} \alpha_t^{(k)} \mathbf{h}_t^{(k)}. \tag{4}$$

Passing $\mathbf{s}_t$ through a fully connected layer with softmax activation results in the final label posterior vector,

$$p(\mathbf{l}|\mathbf{x}_1^t) = \text{softmax}(\mathbf{z}_t), \tag{5}$$

where $\mathbf{z}_t = \mathbf{W}_o\mathbf{s}_t + \mathbf{b}_o$ is the vector of logits, $\mathbf{l}$ is the vector of labels and $(\mathbf{W}_o, \mathbf{b}_o)$ are the weight and bias parameters of the softmax layer.

2.2. Proposed RADMMs

2.2.1. RADMM With Hidden Features (RADMM:H)

One problem with the baseline RADMM is that the mixer LSTM produces mixture weights based on $\mathbf{x}_t$. Thus, it is agnostic to the state of the individual experts. One way to incorporate expert knowledge into the mixer LSTM is to feed it with hidden outputs of the expert LSTMs. This can be achieved by stacking the hidden outputs (features) of each expert and feeding the stacked vector to the mixer LSTM. In the preceding equations, Eq. (2) is replaced by Eq. (6) as

$$(\mathbf{h}_t^{(\text{mix})}, \mathbf{c}_t^{(\text{mix})}) = \text{LSTM}_{\text{mix}}(\widetilde{\mathbf{h}}_t, \mathbf{h}_{t-1}^{(\text{mix})}, \mathbf{c}_{t-1}^{(\text{mix})}), \tag{6}$$

where,

$$\widetilde{\mathbf{h}}_t = \begin{bmatrix} \mathbf{h}_t^{(1)} \\ \mathbf{h}_t^{(2)} \\ \vdots \\ \mathbf{h}_t^{(K)} \end{bmatrix}. \tag{7}$$

With $\mathbf{h}_t^{(\text{mix})}$ known, mixture weights are determined using Eq. (3)-(5).

2.2.2. RADMM With Row Convolution (RADMM:RC)

It is possible to further improve RADMM by projecting the hidden features $\mathbf{h}_t^{(k)}$ of the expert LSTMs to a common subspace. Since inputs to the domain dependent experts undergo different transformations, it is likely that the hidden features of the experts reside on different subspaces. To alleviate this problem, we use linear transforms, one for each expert, to project the hidden features to a common subspace. This is given by

$$\mathbf{g}_t^{(k)} = \mathbf{W}_k\mathbf{h}_t^{(k)}. \tag{8}$$

The projected features can now be stacked column-wise to form $\widetilde{\mathbf{g}}_t = [\mathbf{g}_t^{(1)}\mathbf{g}_t^{(2)} \cdots \mathbf{g}_t^{(K)}]^{\text{T}}$. This is fed to the LSTM mixer as,

$$(\mathbf{h}_t^{(\text{mix})}, \mathbf{c}_t^{(\text{mix})}) = \text{LSTM}_{\text{mix}}(\widetilde{\mathbf{g}}_t, \mathbf{h}_{t-1}^{(\text{mix})}, \mathbf{c}_{t-1}^{(\text{mix})}). \tag{9}$$

The difference between Eq. (9) and Eq. (2) (or Eq. (6)) is the use of input $\widetilde{\mathbf{g}}_t$ instead of $\mathbf{x}_t$ (or $\widetilde{\mathbf{h}}_t$). Following this, the expert weights are determined as usual using Eq. (3). Instead of $\mathbf{h}_t^{(k)}$, $\mathbf{g}_t^{(k)}$ from the experts are linearly combined. Therefore, instead of Eq. (4), Eq. (10) is used and is given by,

$$\mathbf{s}_t = \sum_{k=1}^{K} \alpha_t^{(k)} \mathbf{g}_t^{(k)}. \tag{10}$$

Finally, label posteriors are evaluated using Eq. (5).

2.2.3. RADMM With Component Weighting (RADMM:COM)

So far we have restricted ourselves to improving the inputs of the mixer LSTM. However, it is possible to improve the outputs by introducing component-wise (or element-wise) weighting of the outputs from the experts. This means instead of using scalar weight $\alpha_t^{(k)}$ for each expert, we use a vector of weights in $\boldsymbol{\alpha}_t^{(k)}$. This can be attained as follows. Assume $\mathbf{h}_t^{(\text{mix})}$ is available from the mixer LSTM. A scoring vector $\mathbf{e}_t^{(k)} \in \mathbb{R}^J$ is generated for each expert after passing $\mathbf{h}_t^{(\text{mix})}$ through an affine transform $(\mathbf{V}_k, \mathbf{b}_k)$, one for each expert, using,

$$\mathbf{e}_t^{(k)} = \mathbf{V}_k\mathbf{h}_t^{(\text{mix})} + \mathbf{b}_k. \tag{11}$$

A $J \times K$ scoring matrix $\mathbf{E}$ is then constructed by stacking the scoring vectors column-wise,

$$\mathbf{E} = \begin{bmatrix} | & | & & | \\ \mathbf{e}_t^{(1)} & \mathbf{e}_t^{(2)} & \dots & \mathbf{e}_t^{(K)} \\ | & | & & | \end{bmatrix}_{J \times K}. \tag{12}$$

Keeping the j^{th} row fixed in $\mathbf{E}$, mixture weights are computed by using softmax normalization across experts. Thus, for the k^{th} expert and the j^{th} component, the mixture weight $\alpha_t^{(k)}(j)$ is computed using,

$$\alpha_t^{(k)}(j) = \frac{\exp(e_t^{(k)}(j))}{\sum_{k'=1}^{K} \exp(e_t^{(k')}(j))}, \quad j = 1, \cdots, J, \tag{13}$$

where $\sum_{k=1}^{K} \alpha_t^{(k)}(j) = 1, \forall j \in \{1, \cdots, J\}$. Now the projected features $\mathbf{g}_t^{(k)}$ are linearly combined component-wise with the weight vector $\boldsymbol{\alpha}_t^{(k)}$ as,

$$\mathbf{s}_t = \sum_{k=1}^{K} \boldsymbol{\alpha}_t^{(k)} \odot \mathbf{g}_t^{(k)}, \tag{14}$$

from which label posteriors are calculated using Eq.(5).

2.3. Hidden Linear Interpolation Mixture Model (HLIMM)

A simple way of combining the hidden outputs from the experts is to learn a distinct weight vector $\mathbf{w}_k$ for each expert. Then the learned vector can be used in component-wise linear combination of the hidden outputs. This is given by,

$$\mathbf{s}_t = \sum_{k=1}^{K} \mathbf{w}_k \odot \mathbf{h}_t^{(k)}. \tag{15}$$

Then $\mathbf{s}_t$ can be used to compute label posteriors using Eq.(5). There are some key differences between RADMM and HLIMM. First, there is no mixer LSTM in HLIMM. Second, the linear combination step in Eq.(4) and Eq. (15) differs in the way the weights are computed. While the weights in the former are constrained to probability values, they remain unconstrained in the latter. As will be shown later, HLIMM performed reasonably well despite its simplicity. However, one problem with the trained weights in HLIMM is that they do not change with the change in input test data. Thus, these

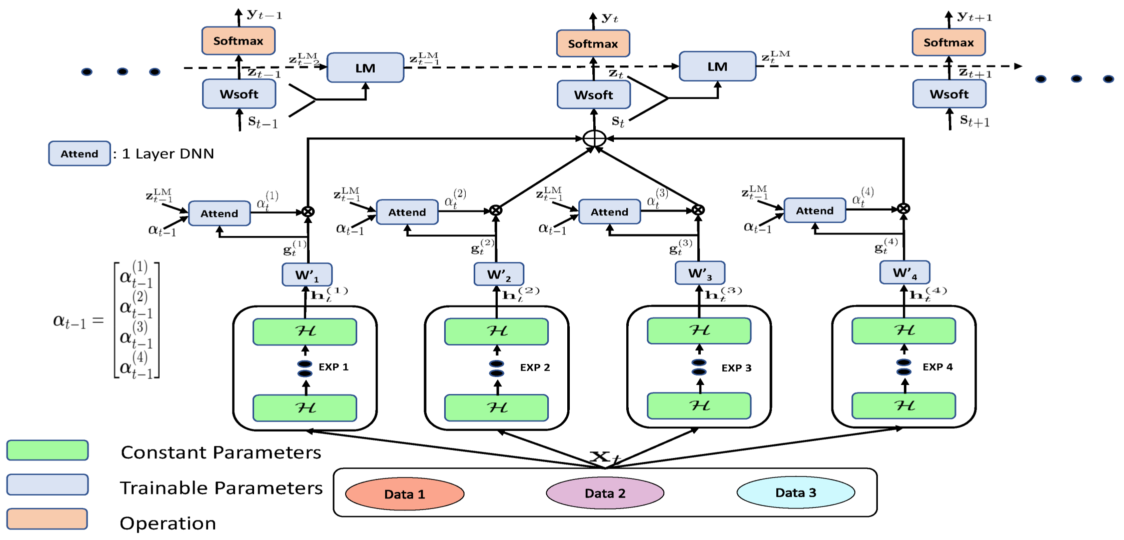

Fig. 1. An example of a Hybrid Attention Mixture Model (HAMM) with 4 experts ($K = 4$).

weights do not generalize well for new scenarios. A better way is to generate these weights on-the-fly based on the input test data.

2.4. Hybrid Attention Mixture Model (HAMM)

We present a brief outline of finding mixture weights using an attention model [22–26] illustrated in Fig. 1. Details are described in [25]. The main components are: (a) the generation of context vectors as row convolution (RC) features, and (b) the computation of the weights of the hidden features using an attention mechanism.

First, the context vector $\mathbf{s}_t$ can be computed as an RC feature by convolving the hidden feature $\mathbf{h}_t^{(k)}$ with learnable weight matrices $\mathbf{W}_k$. The generation of the RC features and linearly combining them is similar to Eq. (8) and Eq. (10) respectively. However, the difference lies in the way mixture weights $\alpha_t(k), k = 1, \cdots, K$ are determined. Thus $\mathbf{s}_t$ is given by,

$$\mathbf{s}_t = \sum_{k=1}^{K} \mathbf{W}_k \mathbf{h}_t^{(k)} = \gamma \sum_{k=1}^{K} \alpha_t^{(k)} \mathbf{g}_t^{(k)}. \quad (16)$$

where the second step holds when $\alpha_t(k) = \frac{1}{K}$ and $\gamma = K$. This is the case for uniform weighting of experts. The term γ is an additional scaling factor that benefits training. For non-uniform weighting, an Attend(.) network is used to learn the mixture weights (attention weights) using,

$$\alpha_t^{(k)} = \text{Attend}(\mathbf{z}_{t-1}, \boldsymbol{\alpha}_{t-1}, \mathbf{g}_t^{(k)}), \quad (17)$$

where $\mathbf{z}_{t-1}$ and $\boldsymbol{\alpha}_{t-1}$ are the logits vector (see Eq. (5)) and mixture weight vector from the previous time step. The Attend(.) function consists of two parts - a scoring function Score(.) followed by normalization. The Score(.) function is a single layer DNN given by,

$$e_t^{(k)} = \text{Score}(\mathbf{z}_{t-1}, \boldsymbol{\alpha}_{t-1}, \mathbf{g}_t^{(k)}), \quad (18)$$

$$= \begin{cases} \mathbf{v}^T \tanh(\mathbf{V}'\mathbf{f}_t + \mathbf{W}'\mathbf{g}_t + \mathbf{b}'), \text{ (location)} \\ \mathbf{v}^T \tanh(\mathbf{U}'\mathbf{z}_{t-1} + \mathbf{W}'\mathbf{g}_t + \mathbf{V}'\mathbf{f}_t + \mathbf{b}') \text{ (hybrid)} \end{cases} \quad (19)$$

where $\mathbf{f}_t = \mathbf{F} * \boldsymbol{\alpha}_{t-1}$ and $*$ is the convolution operation. The first equation of Eq. (19) represents location ($\boldsymbol{\alpha}_{t-1}$) information whereas the second equation represents hybrid attention (HA) since it encodes both content ($\mathbf{z}_{t-1}$) and location information. Selecting either location or hybrid is left to the user. Mixture weights are generated after normalizing the scores using,

$$\alpha_t^{(k)} = \frac{\exp(e_t^{(k)})}{\sum_{k'=1}^{K} \exp(e_t^{(k')})}, \; k = 1, \cdots, K \quad (20)$$

The performance of the attention model in Eq. (17) can be improved further by providing content information that is more reliable than $\mathbf{z}_{t-1}$. This is possible by introducing another recurrent network that utilizes content from several time steps in the past instead of only one step (refer LM block in Fig. 1). This network, in essence, learns a pseudo language model (LM) and can be represented as,

$$\mathbf{z}_{t-1}^{\text{LM}} = \mathcal{H}(\mathbf{u}_{t-1}, \mathbf{z}_{t-2}^{\text{LM}}), \quad \mathbf{u}_{t-1} = \begin{bmatrix} \mathbf{z}_{t-1} \\ \mathbf{s}_{t-1} \end{bmatrix}, \quad (21)$$

$$\boldsymbol{\alpha}_t = \text{Attend}(\mathbf{z}_{t-1}^{\text{LM}}, \boldsymbol{\alpha}_{t-1}, \mathbf{g}), \quad (22)$$

where $\mathcal{H}(.)$ is a LSTM unit. Additional gains can be achieved using component-wise attention which is similar to the ideas in Sec. 2.2.3.

3. EXPERIMENTS

We experimented using transcribed data collected from a wide variety of acoustic conditions on Microsoft devices. These include speech collected from Cortana, conversations, meetings etc. We categorized the test data into two groups - seen and unseen. For the seen group (S1-S4), the acoustic conditions during training and testing were identical. For the unseen group (U1-U4), there was a mismatch in the acoustic conditions between training and testing.

We trained 4 experts (hence, $K = 4$), one per condition, using 30,000 hours (h) of data (cumulative) across S1-S4. Each expert is a 6-layer uni-directional LSTMs [27–30] trained with frame-wise cross-entropy criterion. The LSTMs were equipped with 1024 memory cells in each layer and the cell outputs were linearly projected to 512 dimensions. Then, keeping the parameters of the expert LSTMs constant, we trained the UAMs using only 300 h of data (1% of expert data) in S1-S4. For component-wise weighting in Sec. 2.2.3, we used $J = 64$. The input feature is 80-dimension log Mel filter bank. We applied frame skipping [31] to reduce the run-time cost. The language model is a 5-gram trained using 100 million (M) ngrams.

Word error rates (WERs) of the 4 expert models tested in S1-S4 and U1-U4 are presented in Table 1. For the seen cases S1-S4, the best model was the one whenever there was a match between the training and test conditions. This is expected and is clear from the diagonal pattern highlighted in bold. For the unseen cases U1-U4, the WERs are relatively worse due to the mismatch between training

Table 1. WERs of expert models in various seen and unseen test conditions. Expert models Sn, $n = 1, \cdots, 4$ are named after the acoustic conditions they were trained in. The best expert model (S_{min}) is in the last column. WERs in bold in each row is the lowest WER in that row.

Test	Word	Expert Models				
Cond. ↓	Count	S1	S2	S3	S4	S_{min}
S1	34150	**11.45**	12.91	16.71	18.71	11.45
S2	32134	13.04	**6.97**	23.83	23.14	6.97
S3	46471	19.73	50.12	**14.10**	15.7	14.10
S4	21690	30.87	51.66	21.51	**20.69**	20.69
U1	22618	**16.39**	17.36	22.64	23.39	16.39
U2	29480	37.90	**19.70**	33.60	25.84	19.7
U3	28553	14.91	32.37	14.54	**14.47**	14.47
U4	14715	**13.69**	18.60	15.55	19.99	13.69

and test conditions. However, we can infer that the best performing model is the one whose train condition was closest to the test condition. For e.g., expert model (S1) was trained in a condition that is acoustically closer to the conditions in U1 and U4. Finally, in the last column, we list the WER of the best expert model (S_{min}). Given a test condition, S_{min} is an oracle that picks the best performing expert among the ensemble of experts.

In Table 2, we present the weighted average WER for various UAMs tested separately under three conditions - seen (S1-S4), unseen (U1-U4), and combined (S1-S4, U1-U4). For example, in the first column, we computed the weighted average WER of a UAM by computing the WER in each condition S1 through S4, then weighting each WER by the word count factor (in Table 1), and finally summing the weighted WERs.

In the first column (S1-S4), RADMM:H outperformed the baseline. This proves that making the mixer LSTM state-aware benefits UAM. A sharp drop in WER when using HLIMM indicates that directly learning the vector weights without using a mixer LSTM is also an effective way to model UAMs. We also experimented with learning scalar weights. However, it did not consistently improve over the baseline model. Adding row convolution (RADMM:RC) and component-wise weighting (RADMM:COM) further lowered the WER. RADMM:COM and HLIMM are similar from the perspective that they both use vector weights. However, RADMM:COM outperforming HLIMM is perhaps due to two reasons. First, RADMM:COM generates mixture weights that change with the change in input test data. However, as mentioned in Sec. 2.3, the weights in HLIMM remain constant once training is complete. Second, the mixer LSTM is better at capturing long-term state information than HLIMM. Finally, the most advanced models are the HAMM models outperforming both RADMM and HLIMM models. Within HAMM, HA slightly outperformed LA since both location and content information were included in learning the attention weights. Comparing HAMM:HA with S_{min} models, it is clear that HAMM:HA model has achieved an oracle-like performance. Likewise, in the second column (U1-U4), most notable is the performance of HAMM models since they easily outperform RADMM, HLIMM, and S_{min} models. In the third column (combined) again, the best performing models are HAMM based models.

In Table 3, we outline further gains achieved by HAMM:HA model when additional data were added during training. We observed gains when training using three times (3x) larger data. We also obtained gains when training with additional 10 hours new domain data (car) to teach the training how to handle unseen data. This also proves the following. To train experts, we require large amounts of transcribed data in a wide variety of conditions which may not be always feasible. However, in the absence of experts, the UAMs can be effective even if small amounts of data can be added during training from a wide variety of new acoustic conditions. The UAMs are able to learn these new conditions by finding an interpolation between the acoustic conditions prevalent in experts. Finally, in Table 4, we summarize the performance of the baseline (RADMM:R) and the best proposed model (HAMM:HA) from Table 2. On an average, we observed around 20% relative improvement in WER of HAMM:HA model over RADMM:R model consistently in each test condition. Moreover, our best proposed model outperformed the oracle model.

Table 2. Weighted average WERs of various UAMs tested under seen conditions S1-S4, unseen (U1-U4), and combined conditions (S1-S4, U1-U4). The UAMs were trained with 300 hours of training data. Lowest WER in each condition is highlighted in bold.

UAM	Test Condition		
	S1-S4	U1-U4	Combined
RADMM:R (Baseline)	16.24	19.72	17.68
RADMM:H	16.07	18.91	17.25
HLIMM	13.76	16.37	14.84
RADMM:RC	13.08	16.19	14.37
RADMM:COM	12.92	16.40	14.36
HAMM:LA	12.92	**15.80**	14.11
HAMM:HA	**12.81**	15.82	**14.06**
S_{min} (Oracle)	12.79	16.42	14.30

Table 3. Weighted average WERs of HAMM based UAMs when training with additional data. Lowest WER in each condition is highlighted in bold.

UAM	Data	Test Condition		
		S1-S4	U1-U4	Combined
HAMM:HA+3x	900 h	**12.68**	15.71	13.94
HAMM:HA+Car	310 h	12.75	**15.52**	**13.90**

Table 4. Summary of weighted average WER of UAMs: Baseline (RADMM:R) vs best proposed (HAMM:HA).

Test Cond.	RADMM:R	HAMM:HA	Abs. WER ↓	Rel. WER ↓
S1-S4	16.24	12.81	3.43	21.12
U1-U4	19.72	15.82	3.90	19.78
Combined	17.68	14.06	3.62	20.48

4. CONCLUSIONS AND FUTURE WORKS

We proposed several UAMs using mixture of experts. First, we improved the baseline RADMM model by using a combination of hidden features, row convolution, and vector weights. Second, we introduced a simple UAM which finds mixture weights without using any mixer LSTM. Finally, we use an attention model that uses location and content information to find mixture weights. We evaluated the baseline and proposed methods on speech collected from a wide variety of acoustic conditions on Microsoft devices. Comparing the baseline RADMM and the best proposed HAMM models, we reported relative reduction in WER of about 21.12%, 19,78%, and 20.48% for seen, unseen, and combined conditions respectively. Moreover, the HAMM models were able to beat an oracle model which picks the best expert for a given test condition.

Because the model size of UAMs is several times larger than that of individual models, we are investigating to use teacher-student learning [32] which is proven successful in our large scale tasks [33] to generate a student model with similar size as individual experts while maintaining the power of UAMs. We are also replacing the expert LSTM models with recently proposed layer trajectory LSTM models [34, 35] to make individual experts stronger, resulting even better UAMs.

5. REFERENCES

[1] G. Hinton, L. Deng, D. Yu, G. E. Dahl, A.-R. Mohamed, N. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, T. N. Sainath, et al., "Deep Neural Networks for Acoustic Modeling in Speech Recognition: The Shared Views of Four Research Groups," *IEEE Sig. Process. Magazine.*, vol. 29, no. 6, pp. 82–97, 2012.

[2] D. Yu and J. Li, "Recent Progresses in Deep Learning Based Acoustic Models," *IEEE/CAA J. of Autom. Sinica.*, vol. 4, no. 3, pp. 399–412, July 2017.

[3] G. E. Dahl, D. Yu, L. Deng, and A. Acero, "Context-Dependent Pre-Trained Deep Neural Networks for Large-Vocabulary Speech Recognition," *IEEE Trans. Audio, Speech, Lang. Process.*, vol. 20, no. 1, pp. 30–42, Jan 2012.

[4] O. Hamid, A.-R. Mohamed, H. Jiang, L. Deng, G. Penn, and D. Yu, "Convolutional Neural Networks for Speech Recognition," *IEEE Trans. Audio, Speech, Lang. Process.*, vol. 22, no. 10, pp. 1533–1545, Oct 2014.

[5] S. Hochreiter and J. Schmidhuber, "Long Short-term Memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, Nov 1997.

[6] O. Vinyals, S. V. Ravuri, and D. Povey, "Revisiting Recurrent Neural Networks for Robust ASR," in *ICASSP*, 2012.

[7] D. Yu, M. L. Seltzer, J. Li, J-T. Huang, and F. Seide, "Feature Learning in Deep Neural Networks - Studies on Speech Recognition Tasks," in *Int. Conf. Learn. Rep.*, 2013.

[8] R. A. Jacobs, M. I. Jordan, S. J. Nowlan, and G. E. Hinton, "Adaptive Mixture of Local Experts," *Neural Computation*, vol. 3, no. 1, pp. 79–87, 1991.

[9] J. Tani and S. Nolfi, "Learning to Perceive the World as Articulated: An Approach for Hierarchical Learning in Sensory-Motor Systems," *Neural Networks*, vol. 12, no. 7, pp. 1131–1141, 1999.

[10] M. I. Jordan and R. A. Jacobs, "Hierarchical Mixture Of Experts and the EM Algorithm," *Neural Computation*, vol. 6, no. 2, pp. 181–214, 1994.

[11] D. Eigen, M. A. Ranzato, and I. Sutskever, "Learning Factored Representations in a Deep Mixture of Experts," in *Proc. ICLR*, 2014.

[12] J. Li, L. Deng, Y. Gong, and R. Haeb-Umbach, "An Overview of Noise-Robust Automatic Speech Recognition," *IEEE Trans. Audio, Speech, Lang. Process.*, vol. 22, no. 4, pp. 745–777, Apr. 2014.

[13] S. H. Mallidi, T. Ogawa, K. Veselý, P. S. Nidadavolu, and H. Hermansky, "Autoencoder Based Multi-stream Combination for Noise Robust Speech Recognition," in *Proc. Interspeech*, 2015.

[14] J. Li, L. Deng, R. Haeb-Umbach, and Y. Gong, *Robust Automatic Speech Recognition: A Bridge to Practical Applications*, Academic Press, 2015.

[15] S. E. Chazan, G. Goldberger, and S. Gannot, "Deep Recurrent Mixture of Experts for Speech Enhancement," in *Proc. WASPAA*, 2017.

[16] Y-B. Kim, K. Stratos, and D. Kim, "Domain Attention With an Ensemble of Experts," in *Proc. ACL*, 2017.

[17] M. J. F. Gales, "Cluster Adaptive Training of Hidden Markov Models," *IEEE Trans. Speech, Audio, Process.*, vol. 8, no. 4, pp. 417–428, July 2000.

[18] Kai Yu and M. J. F. Gales, "Discriminative Cluster Adaptive Training," *IEEE Trans. Audio, Speech, Lang. Process.*, vol. 14, no. 5, pp. 1694–1703, Sept 2006.

[19] Tian Tan, Yanmin Qian, Maofan Yin, Yimeng Zhuang, and Kai Yu, "Cluster adaptive training for deep neural network," in *Acoustics, Speech and Signal Processing (ICASSP), IEEE International Conference on*. IEEE, 2015, pp. 4325–4329.

[20] Chunyang Wu and Mark JF Gales, "Multi-basis adaptive neural network for rapid adaptation in speech recognition," in *Acoustics, Speech and Signal Processing (ICASSP), IEEE International Conference on*. IEEE, 2015, pp. 4315–4319.

[21] K. Irie, S. Kumar, M. Nirschl, and H. Liao, "RADMM: Recurrent Adaptive Mixture Model with Applications to Domain Robust Language Modeling," in *Proc. ICASSP*, 2018.

[22] D. Bahdanau, K. Cho, and Y. Bengio, "Neural Machine Translation by Jointly Learning to Align and Translate," in *ICLR*, 2015.

[23] J. Chorowski, D. Bahdanau, D. Serdyuk, K. Cho, and Y. Bengio, "Attention-Based Models for Speech Recognition," in *Conf. on Neural Information Processing Systems*, 2015.

[24] D. Bahdanau, J. Chorowski, D. Serdyuk, P. Brake, and Y. Bengio, "End-to-End Attention-Based Large Vocabulary Speech Recognition," *CoRR*, vol. abs/1508.04395, 2015.

[25] A. Das, J. Li, R. Zhao, and Y. Gong, "Advancing Connectionist Temporal Classification With Attention Modeling," in *Proc. ICASSP*, 2018.

[26] J. Li, G. Ye, A. Das, R. Zhao, and Y. Gong, "Advancing Acoustic-to-Word CTC Model," in *Proc. ICASSP*, 2018.

[27] H. Sak, A. Senior, and F. Beaufays, "Long Short-term Memory Recurrent Neural Network Architectures for Large Scale Acoustic Modeling," in *Proc. Interspeech*, 2014, pp. 338–342.

[28] A. Graves, A. Mohamed, and G. Hinton, "Speech recognition with deep recurrent neural networks," in *Proc. ICASSP*, 2013, pp. 6645–6649.

[29] Y. Miao and F. Metze, "On speaker adaptation of long short-term memory recurrent neural networks.," in *Proc. Interspeech*, 2015, pp. 1101–1105.

[30] Xiangang Li and Xihong Wu, "Constructing long short-term memory based deep recurrent neural networks for large vocabulary speech recognition," in *Proc. ICASSP*, 2015, pp. 4520–4524.

[31] Y. Miao, J. Li, Y. Wang, S. Zhang, and Y. Gong, "Simplifying Long Short-term Memory Acoustic Models for Fast Training AND Decoding," in *Proc. ICASSP*, 2016.

[32] Jinyu Li, Rui Zhao, Jui-Ting Huang, and Yifan Gong, "Learning small-size DNN with output-distribution-based criteria.," in *Proc. Interspeech*, 2014, pp. 1910–1914.

[33] J. Li, R. Zhao, Z. Chen, et al., "Developing far-field speaker system via teacher-student learning," in *Proc. ICASSP*, 2018.

[34] Jinyu Li, Changliang Liu, and Yifan Gong, "Layer trajectory LSTM," in *Proc. Interspeech*, 2018.

[35] Jinyu Li, Liang Lu, Changliang Liu, and Yifan Gong, "Improving layer trajectory LSTM with future context frames," in *Proc. ICASSP*, 2019.