A Unified Speaker Adaptation Approach for ASR

Yingzhu Zhao[1,2*], **Chongjia Ni**[2], **Cheung-Chi Leung**[2]
Shafiq Joty[1], **Eng Siong Chng**[1], **Bin Ma**[2]
[1]Nanyang Technological University, Singapore
[2]Machine Intelligence Technology, Alibaba Group
{srjoty, aseschng}@ntu.edu.sg
{yingzhu.zhao, ni.chongjia, cc.leung, b.ma}@alibaba-inc.com

Abstract

Transformer models have been used in automatic speech recognition (ASR) successfully and yields state-of-the-art results. However, its performance is still affected by speaker mismatch between training and test data. Further finetuning a trained model with target speaker data is the most natural approach for adaptation, but it takes a lot of compute and may cause *catastrophic forgetting* to the existing speakers. In this work, we propose a unified speaker adaptation approach consisting of feature adaptation and model adaptation. For feature adaptation, we employ a speaker-aware persistent memory model which generalizes better to unseen test speakers by making use of speaker i-vectors to form a persistent memory. For model adaptation, we use a novel gradual pruning method to adapt to target speakers without changing the model architecture, which to the best of our knowledge, has never been explored in ASR. Specifically, we gradually prune less contributing parameters on model encoder to a certain sparsity level, and use the pruned parameters for adaptation, while freezing the unpruned parameters to keep the original model performance. We conduct experiments on the Librispeech dataset. Our proposed approach brings relative 2.74-6.52% word error rate (WER) reduction on general speaker adaptation. On target speaker adaptation, our method outperforms the baseline with up to 20.58% relative WER reduction, and surpasses the finetuning method by up to relative 2.54%. Besides, with extremely low-resource adaptation data (e.g., 1 utterance), our method could improve the WER by relative 6.53% with only a few epochs of training.

1 Introduction

End-to-end models yield state-of-the-art performance on automatic speech recognition (ASR) in the past decade, such as connectionist temporal classification (CTC) model (Miao et al., 2015; Graves, 2012), attention-based encoder-decoder model (Zhang et al., 2017), recurrent neural network transducer (RNN-T) (Graves, 2012), transformer model (Dong et al., 2018) and conformer model (Gulati et al., 2020). However, model performance deteriorates due to *speaker mismatch* between training and test data. Given the target speaker, finetuning the trained model could alleviate the speaker mismatch problem to some extent, but finetuning the entire model requires large amounts of compute to be effective, and it could in turn bring catastrophic forgetting (McCloskey and Cohen, 1989) to the existing speakers.

Currently, there are two lines of studies to address the speaker mismatch problem in neural network based models. One category is working on the acoustic features, i.e., either by normalizing acoustic features to be speaker-independent (Seide et al., 2011; Tomashenko and Estève, 2018; Ochiai et al., 2018) or by introducing additional speaker related knowledge (e.g., i-vector) to adapt the acoustic model (Saon et al., 2013; Senior and Lopez-Moreno, 2014; Pan et al., 2018; Fan et al., 2019). A summary vector of each utterance can be trained to replace speaker i-vector (Vesely et al., 2016). To adapt to acoustic variability, Kim et al. (2017) add shifting and scaling parameters in the layer-normalization layer.

The other category belongs to model adaptation, i.e., to train the speaker-dependent model from speaker-independent model parameters with extra adaptation data. To avoid overfitting, techniques such as L2 regularization (Liao, 2013), Kullback-Leibler divergence (Yu et al., 2013) and adversarial multitask learning (Meng et al., 2019) have been used. Because finetuning the entire model is computationally expensive, Yao et al. (2012); Siniscalchi et al. (2013); Samarakoon and Sim (2016a) only adapt specific layers or a subset of parame-

*Yingzhu Zhao is under the Joint PhD Program between Alibaba and Nanyang Technological University.

Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing, pages 9339–9349
November 7–11, 2021.

ters. In particular, Swietojanski et al. (2016); Samarakoon and Sim (2016b); Xie et al. (2019) reparameterize each hidden unit with speaker-dependent amplitude function in fully-connected or convolutional neural network layers. However, it is difficult to determine which model parameters to adapt for target speaker, and choosing certain sub-layer(s) intuitively may not be optimal.

In this work, we propose a unified speaker adaptation model by making use of both feature adaptation and model adaptation. For feature adaptation, we propose the speaker-aware persistent memory model to generalize better to unseen test speakers. In particular, speaker i-vectors from the training data are sampled and concatenated to speech utterances in each encoder layer, and the speaker knowledge is learnt through attention computation with speaker i-vectors. Our method learns utterance level speaker knowledge, which is more effective than learning time step dependent speaker knowledge (Fan et al., 2019) since it is more robust to various variability factors along an utterance.

For model adaptation, we explore gradual pruning (Zhu and Gupta, 2018), which to the best of our knowledge, is the first time being studied for speaker adaptation. We gradually prune less contributing parameters on model encoder, and then use the pruned parameters for target speaker adaptation while freeze the unpruned parameters to retain the model performance on general speaker data. In this way, our model could adapt to target speakers very fast by updating only a small percentage (10%) of encoder parameters, and it does not change the model architecture. Freezing unpruned parameters alleviates the catastrophic forgetting problem as well.

Our proposed approach brings relative 2.74-6.52% WER reduction on general speaker adaptation. On target speaker adaptation, our method outperforms the baseline with up to 20.58% relative WER reduction, and surpasses the finetuning method by up to relative 2.54%[1].

2 Background

2.1 Speech Transformer

Speech transformer (Dong et al., 2018) is an extension of the transformer model (Vaswani et al., 2017) for ASR. We briefly introduce the speech transformer model here. For a speech input sequence, speech transformer first applies two convolution layers with stride two to reduce hidden representation length. A sinusoidal positional encoding is added to encode position information. Both encoder and decoder in speech transformer model use multi-head attention network. Attention network has three inputs key, query and value, which are distinct transformations of an input sequence. The multi-head attention network is computed by concatenating single attention network h times:

$$Attention(Q, K, V) = softmax(\frac{QK^T}{\sqrt{d_k}})V \quad (1)$$

$$MultiHd(Q, K, V) = Concat(hd_1, ..., hd_h)W^O \quad (2)$$

$$hd_i = Attention(QW_i^Q, KW_i^K, VW_i^V) \quad (3)$$

where h is the head number, $W_i^Q \in \mathbb{R}^{d_{model} \times d_q}$, $W_i^K \in \mathbb{R}^{d_{model} \times d_k}$, $W_i^V \in \mathbb{R}^{d_{model} \times d_v}$, $W^O \in \mathbb{R}^{hd_v \times d_{model}}$, we set $d_k = d_q = d_v = d_{model}/h$.

Multi-head attention could learn input representation in different subspaces simultaneously. For encoder, the three inputs all come from the speech input, so the attention network is called self-attention network. For decoder, the text input first goes through self-attention network. To maintain autoregression in decoder, a mask is applied to future tokens. To incorporate information from the speech input, in the next attention network, key and value vectors come from encoder, and query vector comes from decoder, so this attention network is called cross-attention network. Layer normalization and residual connection are applied before and after multi-head attention network. Afterwards, there is a position-wise feedforward network with rectified linear unit (ReLU) activation:

$$FFN(x) = max(0, xW_1 + b_1)W_2 + b_2 \quad (4)$$

where $W_1 \in \mathbb{R}^{d_{model} \times d_{ff}}$, $W_2 \in \mathbb{R}^{d_{ff} \times d_{model}}$, and the biases $b_1 \in \mathbb{R}^{d_{ff}}$, $b_2 \in \mathbb{R}^{d_{model}}$.

Self-attention network and position-wise feedforward network form an encoder layer. There is an additional cross-attention network in a decoder layer. There are N_e encoder layers and N_d decoder layers in total.

2.2 Speaker Adaptation

Speaker adaptation arises due to *speaker mismatch* between training and test data. It aims to adapt the

[1]Our code is available at https://github.com/zyzpower/gradprune_speaker

model to a target speaker, which is a critical component in HMM-based models (Keith and Matthias, 2005; Furui, 1980; Gauvain and Lee, 1994; Kuhn et al., 2000). For neural network based models, many approaches are developed as well as discussed briefly in Section 1.

Adapting an ASR model is a challenging task given that ASR model is a huge and complex model with a large number of parameters to update. Fine-tuning the entire model takes significant computational resources to reach the optimal performance, and it potentially causes catastrophic forgetting problem (McCloskey and Cohen, 1989; Kirkpatrick et al., 2017), which means when model parameters trained for existing speakers are adapted for target speaker, knowledge learnt previously is lost.

2.3 I-vector

I-vector is a low-dimension vector that is dependent on speaker and channel. I-vector dimension is fixed as defined no matter how long the utterance is. It is extracted based on a data-driven approach by mapping frames of an utterance to a low-dimensional vector space using a factor analysis technique (Dehak et al., 2011a). The system is based on Support Vector Machines or directly using the cosine distance value as a final decision score (Dehak et al., 2011b). I-vector was initially invented for audio classification and identification, but recently it is used for speaker adaptation as well (Saon et al., 2013; Dehak et al., 2011b; Karafiat et al., 2011).

3 Proposed Method

We propose an efficient speaker adaptation model by making use of both feature adaptation and model adaptation. For feature adaptation, we embed speaker knowledge represented by a number of fixed speaker i-vectors into each input utterance (Zhao et al., 2020). This aims to capture speaker information through attention computation between each utterance and speaker i-vectors. For model adaptation, an effective method is employed that can adapt to target speaker very fast without sacrificing performance on existing speakers. In particular, we prune the model gradually and finetune a small subset of parameters to be speaker-specific.

3.1 Speaker-Aware Persistent Memory for Feature Adaptation

Speaker-aware persistent memory model learns speaker knowledge from i-vectors. We first randomly sample N speaker i-vectors $m_1, ..., m_N \in \mathbb{R}^{d_k}$ which form the speaker space (Gales, 1998; Yu and Gales, 2006). Here we have the assumption that the linear combinations of speaker space are enough to cover the speaker information space, i. e., any unknown speaker not seen in the training data can be represented approximately by the sampled i-vectors from the training data. We name the learned transformation of speaker space as persistent memory vectors M_k and M_v:

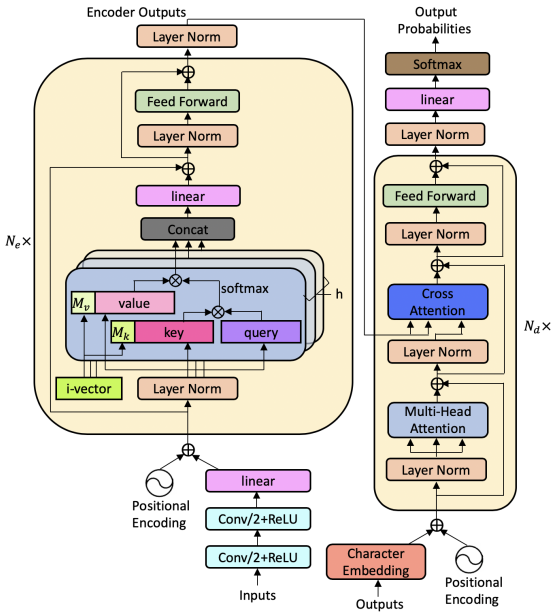

Figure 1: Speaker-aware persistent memory model. M_k and M_v from speaker i-vectors are concatenated to key and value vectors.

$$M_k = Concat([U_k m_1, ..., U_k m_N]) \in \mathbb{R}^{N \times d_k} \quad (5)$$

$$M_v = Concat([U_v m_1, ..., U_v m_N]) \in \mathbb{R}^{N \times d_k} \quad (6)$$

where $U_k \in \mathbb{R}^{d_k \times d_k}$, $U_v \in \mathbb{R}^{d_k \times d_k}$. Only U_k and U_v matrices are learnable while sampled i-vectors are fixed in this method.

With the persistent memory vectors, we concatenate them respectively to the input vectors of self-attention network $X = [x_1, ..., x_t]$ to be the new key and value vectors. Attention network thus captures speaker-specific knowledge through attention computation between each utterance and persistent memory vectors as Eq. 9:

$$K_m = [k_1, ..., k_{t+N}] = ([W_k x_1, ..., W_k x_t], M_k) \quad (7)$$

$$V_m = [v_1, ..., v_{t+N}] = ([W_v x_1, ..., W_v x_t], M_v) \tag{8}$$

$$Attention(Q, K_m, V_m) = softmax(\frac{QK_m^T}{\sqrt{d_k}})V_m \tag{9}$$

Since M_k and M_v are shared across all layers, they form the persistent memory. Given that persistent memory is meant for capturing speaker knowledge, we name it as speaker-aware persistent memory. The overall framework of speaker-aware persistent memory model is shown in Figure 1.

Since our method aims at learning any speaker information from the speaker space, it effectively addresses the problem of having unseen speakers in the test data. Furthermore, using static i-vectors saves the effort to compute the i-vectors of all speakers in the training data. Besides, the attention computation (Eq. 9) with persistent memory vectors is taken over the entire utterance x_1 to x_t, so each input speech time step takes part in extracting speaker information. This holistic consideration is more effective than Fan et al. (2019) who compute time step dependent speaker representations, which may be more susceptible to various variability factors along an utterance such as speaking rhythm.

3.2 Gradual Pruning for Model Adaptation

The speaker-aware persistent memory method discussed above is for general speaker adaptation without knowing target speaker profile. If the target speaker data is available, finetuning the trained model with target speaker data could result in catastrophic forgetting problem as detailed in Section 2.2. To address this, we take advantage of an effective approach to adapt to target speaker in a fast manner, and retain the model performance on the general speaker data at the same time.

Given an input speech $y = \{y_1, ..., y_n\}$ and output text $z = \{z_1, ..., z_m\}$, ASR models the conditional probability of the output text over the input speech as follows:

$$P(z|y;\theta) = \prod_{i=1}^{m} P(z_i|y, z_{<i};\theta) \tag{10}$$

where θ represents the model parameters. Given a training dataset $D = \{y_D^j, z_D^j\}_{j=1}^{L}$, θ is trained to maximize the following log-likelihood objective:

$$\mathcal{J}(\theta) = \underset{\theta}{\text{argmax}} \sum_{j=1}^{L} \log P(z_D^j|y_D^j;\theta) \tag{11}$$

Given the target speaker dataset $D_t = \{y_{D_t}^j, z_{D_t}^j\}_{j=1}^{L_t}$, directly finetuning the trained model means continuing to train the model to maximize the log-likelihood:

$$\mathcal{J}(\theta_{D_t}) = \underset{\theta_{D_t}}{\text{argmax}} \sum_{j=1}^{L_t} \log P(z_{D_t}^j|y_{D_t}^j;\theta_{D_t}) \tag{12}$$

where θ_{D_t} is initialized with the trained parameters θ in Eq. 11.

As shown in many recent studies (Frankle and Carbin, 2019; Zhu and Gupta, 2018; Liu et al., 2019), not all parameters in a neural network model contribute to the training objective. Pruning the redundant parameters leads to negligible performance degradation (Li et al., 2017; Han et al., 2015) or may even outperform the original model (Zhu and Gupta, 2018) due to better generalization.

Our experiments in Figure 2 show that up to 50% of encoder parameters can be pruned in ASR with negligible performance degradation. Therefore, we first prune the model with training data gradually to the predetermined sparsity level by zeroing out low magnitude parameters for every 10k training steps, i.e., only retain a certain percentage of high magnitude unpruned parameters θ_{UP} ultimately. This is to unearth the sub-network whose performance well matches the original model.

Different from Liang et al. (2021) who prune on a well-trained model, we train and prune concurrently as seen in Figure 3(b) (with warm-up training first) to reduce the total number of training steps

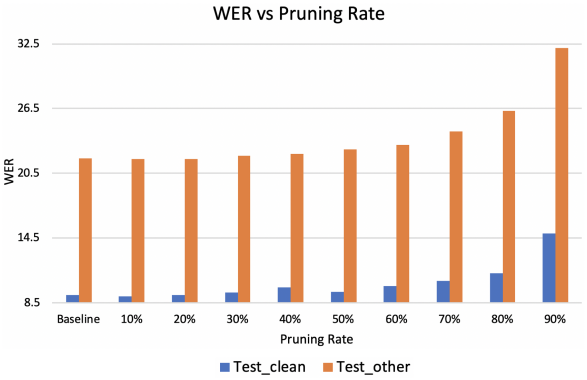

Figure 2: WER results on Librispeech test data with different pruning rates.

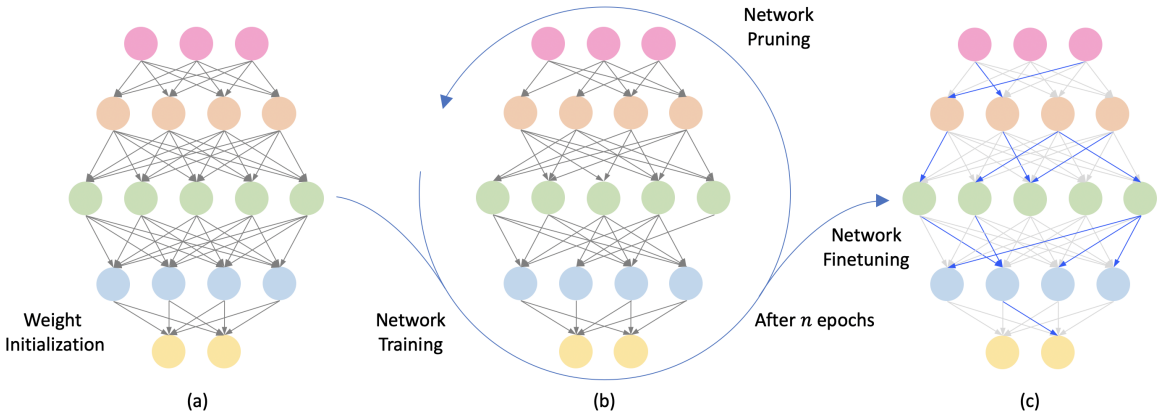

Figure 3: Illustration of gradual pruning method for speaker adaptation. After network parameters are initialized (a), we train and prune the model at the same time for n epochs (b), and lastly finetune the pruned parameters with target speaker data (c). Light gray connections in (c) mean corresponding parameters are frozen, while blue ones indicate parameters are finetuned, which are the parameters pruned at earlier stage (b).

and thus save computation resources. Because speech or speaker information is learnt through the encoder of an end-to-end ASR model, we only prune encoder parameters including embedding network, self-attention network and feedforward network in all encoder layers.

Afterwards, we keep the informative sub-network untouched by freezing the unpruned parameters θ_{UP} to retain the performance on existing speakers, as represented by the light gray connections in Figure 3(c), and only finetune the pruned free parameters θ_P for target speaker adaptation (blue connections in Figure 3(c)). The training objective will be:

$$\mathcal{J}(\theta_P) = \underset{\theta_P}{\operatorname{argmax}} \sum_{j=1}^{L_t} \log P(z_{D_t}^j | y_{D_t}^j; \theta_{UP}, \theta_P) \tag{13}$$

where θ_{UP} is frozen and θ_P is updated. Since the informative sub-network is already capable of performing ASR task very well, we believe further finetuning the free parameters with target speaker data is an added value to the speaker-specific model.

Our method does not change the model architecture, unlike some approaches to attach an additional adapter module (Ding et al., 2020). Besides, we only need to finetune a small number of parameters compared to finetuning the entire model. Fixing the informative sub-network makes our model retain past knowledge with no catastrophic forgetting issue. It also prevents the model from easily overfitting on low-resource target speaker data to some extent.

4 Experiments

In this section, we present our experiments using the proposed speaker-aware persistent memory model and the gradual pruning method.

4.1 Datasets

We conduct experiments to confirm the effectiveness of the proposed model on the open-source Librispeech dataset (Panayotov et al., 2015). LibriSpeech consists of 16kHz read English speech from audiobooks. We use the given train/development/test splits of Librispeech dataset. Test_clean data is clean and Test_other data has noise in speech. See Appendix A.1 for the statistics of Librispeech dataset used in our experiments.

4.2 Training Setup

We use PyTorch and Espnet (Watanabe et al., 2018) toolkit for our experiments, and we train the model for 100 epochs ($n = 100$ in Figure 3(b)). We use the best set of hyperparameters tested by Watanabe et al. (2018) for transformer model without further tuning, and we pre-process the data following the Espnet toolkit. The total number of model parameters is 31 million. Input features are generated by 80-dimensional filterbanks with pitch on each frame, with a window size of 25ms shifted every 10ms. The acoustic features are mean and variance normalized. We exclude utterances longer than 3000 frames or 400 characters to keep memory manageable. For joint decoding of CTC and attention, the coefficient is 0.3 and 0.7 for CTC and attention respectively. The convolutional frontend before transformer encoder is two 2D convolutional

Model	Test_clean	Test_other
End-to-end (E2E) (Lüscher et al., 2019)	14.7	40.8
E2E with augmented data (Bérard et al., 2018)	15.1	-
Local prior matching (Hsu et al., 2020)	14.85	39.95
LAS (Irie et al., 2019)	12.9	35.5
Self-training (Kahn et al., 2020)	8.06	30.44
Baseline	9.2	21.9

Table 1: WER results of speech recognition models on LibriSpeech 100h.

neural network layers with filter size (3,2) and stride 2, each followed by a ReLU activation. The attention dimension d_{model} is 256, and the feedforward network hidden state dimension d_{ff} is 2048. In the transformer structure, the number of attention heads h is 4, with $d_k = d_q = d_v = 64$ for each head, the number of encoder layers N_e is 12, the number of decoder layers N_d is 6, the initial value of learning rate is 5.0, the encoder and decoder dropout rate is 0.1. The input samples are shuffled randomly and trained with batch size 12. We use unigram sub-word algorithm with the vocabulary size capped to be 5000. For i-vector generation, we follow SRE08 recipe in Kaldi (Povey et al., 2011) toolkit on the training data. I-vectors extracted are of dimension 100. They are then transformed to have the same dimension as speech vectors for concatenation. Our baseline model is competitive compared with other model results from Table 1.

4.3 Experimental Results

4.3.1 Adaptation for General Speakers

We first test on the adaptation for general speakers without knowing target speaker profile. Speaker-aware persistent memory model introduced in Section 3.1 achieves this objective. Here we omit the hyperparameter tuning part, and directly use the best hyperparameters tested by Zhao et al. (2020), including the number of speaker i-vectors in the speaker space and the number of layers applied with speaker-aware persistent memory. We randomly sample 64 speaker i-vectors and apply on all the encoder layers in speech transformer. 64 i-vectors were tested to be a good choice to provide diverse speaker information (Zhao et al., 2020), and applying on all encoder layers helps capture speaker knowledge from both low-level phonetic features and high-level global information. Table 2 shows that our method brings 2.74-6.52% relative improvement over the baseline, and surpasses Fan et al. (2019) who also use speaker i-vectors. Furthermore, here we also compare our model with the first persistent memory model used in ASR (You et al., 2019), in which persistent memory vectors are randomly initialized and meant to capture general knowledge. Different from them, our model is to address the speaker mismatch issue. Our method achieves the best results.

Model	Test_clean	Test_other
Baseline	9.2	21.9
You et al. (2019)	8.9	21.6
Fan et al. (2019)	8.9	21.4
Ours	8.6	21.3

Table 2: State-of-the-art results of different speaker adaptation algorithms on Librispeech test data.

4.3.2 Adaptation for Target Speaker

If the target speaker profile is known beforehand, the gradual pruning method discussed in Section 3.2 could adapt to the target speaker. Directly finetuning the entire model takes high computation resources by updating all model parameters, and could overfit easily if the amount of target speaker data is limited. We are interested to see the performance of the gradual pruning method especially on low-resource data, as well as how much it alleviates the catastrophic forgetting problem. Therefore, we randomly choose a speaker from the Librispeech Test_other data as the target speaker, and only select 10 utterances of the target speaker as training data. The remaining utterances of the target speaker are chosen as the test data. We do this four times and report the average performance to see the generalizability of the proposed approach. The average baseline WER of four speakers is 20.5, and is slightly smaller than the average WER of Test_other speakers, which is 21.9, so further improving the target speaker performance is a bit more challenging. The pruning rate is set as 10% here. We compare the perfor-

Figure 4: WER results of target speaker (a) and non-target speakers (b). Finetune: directly finetuning the entire model as Eq. 12. I-vec: speaker-aware persistent memory method proposed in Section 3.1 by adding i-vectors. Pruning: gradual pruning proposed in Section 3.2. Pruning+I-vec: combining feature adaptation and model adaptation methods proposed. The dotted lines are the second order polynomial trendlines.

mance of 1) Finetune: directly finetuning the entire model as Eq. 12, 2) I-vec: speaker-aware persistent memory method by adding i-vectors, 3) Pruning: gradual pruning, 4) Pruning+I-vec: combining feature adaptation and model adaptation methods proposed.

For results on the target speaker in Figure 4(a), finetuning works better than the baseline. Adding i-vectors has the highest WER initially and the performance is worse than simply finetuning the trained model after 20 epochs. We believe speaker-aware persistent memory method works better on general speaker adaptation given that the sampled i-vectors form the speaker space to capture any speaker knowledge. It is not designed to adapt to some specific speakers. Using the gradual pruning method alone has lower WER than finetuning at the initial stage, but surprisingly it overfits more than the finetuning method after 20 epochs. More detailed analysis is needed and we leave it to future work. Lastly, we combine the feature adaptation and model adaptation methods, and it achieves our best result. It outperforms the baseline with up to 20.58% relative WER reduction, and surpasses the finetuning method by up to relative 2.54%. We see that the feature adaptation method and the model adaptation method we propose complement each other, as the combined model result surpasses each individual one.

We want to analyze the performance of the rest non-target speaker data to see if catastrophic forgetting happens. From Figure 4(b), all target speaker adapted models perform slightly worse than the baseline, which is expected. Combining feature adaptation and model adaptation could alleviate catastrophic forgetting problem effectively. It generally outperforms finetuning in Figure 4(b).

5 Analysis

In this section, we revisit our approach to reveal more details and explore the effectiveness of the gradual pruning method in combination with the speaker-aware persistent memory model.

5.1 Pruning Rate

We first test different pruning rates on encoder. Results are shown in Figure 5. Less pruning rate keeps more parameters for the general speaker data, and has less learning capability to target speaker. It is more suitable for simple adaptation tasks. Higher pruning rate generates a more sparse network and is more flexible for speaker adaptation, except that it retains less original model parameters, thus forgets more on the general speaker data. It can be seen from Figure 5 that pruning 10% of encoder parameters achieves the best result.

5.2 Gradual Pruning vs One-time Pruning

We use the gradual pruning method (Zhu and Gupta, 2018) to prune to target sparsity for every 10k training steps. One-time pruning at the initial/middle/final stage of the overall training is tested for comparison as well. We train for 100 epochs, and initial/middle/final stage pruning is done at 0/50/100 epoch respectively. Gradual pruning and one-time pruning will reach the same sparsity level after the training. Here we use either gradual or one-time pruning at different stages during training, and show the best results of finetuning for 15 epochs. Table 3 shows that gradual pruning

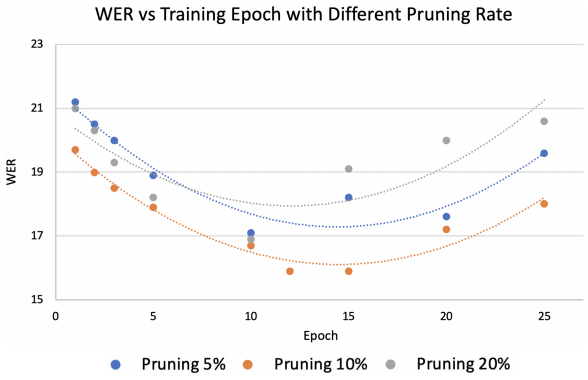

Figure 5: WER results of target speaker with different pruning rates. The dotted lines are the second order polynomial trendlines.

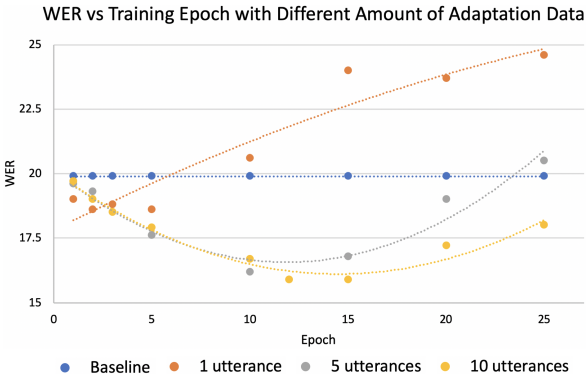

Figure 6: WER results of target speaker with different amount of adaptation data. The dotted lines are the second order polynomial trendlines.

Model	Target Speaker
Baseline	19.9
One-time pruning at initial stage	16.2
One-time pruning at middle stage	17.6
One-time pruning at final stage	16.0
Gradual pruning	15.9

Table 3: WER results of gradual pruning versus one-time pruning. We train for 100 epochs, and initial/middle/final stage pruning is done at 0/50/100 epoch respectively.

works better than one-time pruning, be it initial, middle or final stage of the training. Compared with one-time pruning, gradual pruning could learn and prune at the same time. In particular, gradual pruning follows the train prune cycle, and is capable of iteratively learning the unpruned parameters after less contributing parameters are pruned. For the one-time pruning, pruning at an earlier stage has the advantage to let the model learn the unpruned parameters based on the pruned ones in the remaining training of the model, but pruning earlier has the risk to prune important parameters since the model is not well learnt yet, vice versa for pruning late. Hence, gradual pruning works the best.

5.3 Extremely Low-resource Adaptation Data

Lastly, we would like to see the extremely low-resource adaptation data scenarios. We reduce the amount of adaptation data and compare the performance with the baseline, where no adaptation is performed. The characteristics of the adaptation data selected are listed in Table 4. From Figure 6, when the amount of adaptation data is reduced from 10 utterances to 5 utterances, the results are similar to that of 10 utterances at the initial training stage, and could outperform the baseline by up to relative 18.59%. With less adaptation data, the model overfits much faster, especially in the case of having only 1 utterance for adaptation. However, even with only 1 utterance, it could surpass the baseline by up to relative 6.53% with only 5 epochs of training. Therefore, even with extremely low-resource adaptation data such as 1 utterance, our method is effective with fast adaptation.

Utterance(s)	1	5	10
Total No. of Words	32	104	174
Total Duration (s)	15.00	40.00	67.17

Table 4: Characteristics of utterances selected as the extremely low-resource adaptation data.

6 Conclusion

In this paper, we have proposed a unified speaker adaptation approach consisting of feature adaptation and model adaptation. Speaker-aware persistent memory model makes use of speaker i-vectors to adapt at the feature level, and we use the gradual pruning approach to retrieve a subset of model parameters for adaptation at the model level. Gradual pruning is found to be better than one-time pruning because gradual pruning could iteratively learn based on pruned parameters. It can alleviate catastrophic forgetting problem as well by retaining a subnetwork whose performance matches the original network. We find that our proposed method is effective in both general speaker adaptation and specific target speaker adaptation. In particular, our method brings relative 2.74-6.52% WER reduction on general speaker adaptation, and outperforms the baseline with up to 20.58% relative WER reduction on target speaker adaptation. Even with extremely

low-resource adaptation data, our method could bring 6.53% relative improvement with only a few training epochs. In the future, we are interested in the overfitting issue with low-resource data, as well as multi-speaker adaptation with our method.

References

Alexandre Bérard, Laurent Besacier, Ali Can Kocabiyikoglu, and Olivier Pietquin. 2018. End-to-end automatic speech translation of audiobooks. In *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE.

Najim Dehak, Pedro A.Torres-Carrasquillo, Douglas Reynolds, and Reda Dehak. 2011a. Language recognition via ivectors and dimensionality reduction. In *INTERSPEECH 2011 – 12th Annual Conference of the International Speech Communication Association*, pages 857–860.

Najim Dehak, Patrick J Kenny, Réda Dehak, Pierre Dumouchel, and Pierre Ouellet. 2011b. Front-end factor analysis for speaker verification. *IEEE Transactions on Audio, Speech, and Language Processing*, 19(4):788–798.

Fenglin Ding, Wu Guo, Lirong Dai, and Jun Du. 2020. Attention-based gated scaling adaptive acoustic model for CTC-based speech recognition. In *2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*.

Linhao Dong, Shuang Xu, and Bo Xu. 2018. Speech-transformer: A no-recurrence sequence-to-sequence model for speech recognition. In *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5884–5888. IEEE.

Zhiyun Fan, Jie Li, Shiyu Zhou, and Bo Xu. 2019. Speaker-aware speech-transformer. In *2019 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*, pages 222–229.

Jonathan Frankle and Michael Carbin. 2019. The lottery ticket hypothesis: Finding sparse, trainable neural networks. In *2019 International Conference on Learning Representations (ICLR)*.

S. Furui. 1980. A training procedure for isolated word recognition systems. *IEEE Transactions on Audio, Speech, and Language Processing*, 28(2):129–136.

Mark J. F. Gales. 1998. Cluster adaptive training for speech recognition. *Int. Conf. Speech Language Processing*, 5:1783–1786.

J.-L. Gauvain and Chin-Hui Lee. 1994. Maximum a posteriori estimation for multivariate gaussian mixture observations of markov chains. *IEEE Transactions on Audio, Speech, and Language Processing*, 2(2):291–298.

Alex Graves. 2012. Sequence transduction with recurrent neural networks. In *International Conference of Machine Learning (ICML)*, pages 235–242.

Anmol Gulati, James Qin, Chung-Cheng Chiu, Niki Parmar, Yu Zhang, Jiahui Yu, Wei Han, Shibo Wang, Zhengdong Zhang, Yonghui Wu, and Ruoming Pang. 2020. Conformer: Convolution-augmented transformer for speech recognition. In *INTERSPEECH 2020 – 21st Annual Conference of the International Speech Communication Association*, pages 5036–5040.

Song Han, Jeff Pool, John Tran, and William J. Dally. 2015. Learning both weights and connections for efficient neural networks. In *Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems*.

Wei-Ning Hsu, Ann Lee, Gabriel Synnaeve, and Awni Hannun. 2020. Self-supervised speech recognition via local prior matching. *arXiv preprint arXiv:2002.10336*.

Kazuki Irie, Rohit Prabhavalkar, Anjuli Kannan, Antoine Bruguier, David Rybach, and Patrick Nguyen. 2019. On the choice of modeling unit for sequence-to-sequence speech recognition. In *INTERSPEECH 2019 – 20th Annual Conference of the International Speech Communication Association*.

Jacob Kahn, Ann Lee, and Awni Hannun. 2020. Self-training for end-to-end speech recognition. In *2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE.

Martin Karafiat, Lukas Burget, Pavel Matejka, Ondrej Glembek, and Jan Cernocky. 2011. iVector-based discriminative adaptation for automatic speech recognition. In *2011 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*, pages 152–157.

Johnson Keith and Sjerps Matthias. 2005. Speaker normalization in speech perception. In *The Handbook of Speech Perception*, pages 363–389. Wiley Online Library.

Taesup Kim, Inchul Song, and Yoshua Bengio. 2017. Dynamic layer normalization for adaptive neural acoustic modeling in speech recognition. In *INTERSPEECH 2017 – 18th Annual Conference of the International Speech Communication Association*.

James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A. Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, Demis Hassabis, Claudia Clopath, Dharshan Kumaran, and Raia Hadsell. 2017. Overcoming catastrophic forgetting in neural networks. In *Proceedings of the national academy of sciences*, pages 3521–3526.

Roland Kuhn, Jean-Claude Junqua, Patrick Nguyen, and Nancy Niedzielski. 2000. Rapid speaker adaptation in eigenvoice space. *IEEE Transactions on*

Audio, Speech, and Language Processing, 8(6):695–707.

Hao Li, Asim Kadav, Igor Durdanovic, Hanan Samet, and Hans Peter Graf. 2017. Pruning filters for efficient convnets. In *2017 International Conference on Learning Representations (ICLR)*.

Jianze Liang, Chengqi Zhao, Mingxuan Wang, Xipeng Qiu, and Lei Li. 2021. Finding sparse structures for domain specific neural machine translation. In *The Thirty-Fifth AAAI Conference on Artificial Intelligence, AAAI 2021*.

Hank Liao. 2013. Speaker adaptation of context dependent deep neural networks. In *2013 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 7947–7951.

Zhuang Liu, Mingjie Sun, Tinghui Zhou, Gao Huang, and Trevor Darrell. 2019. Rethinking the value of network pruning. In *2019 International Conference on Learning Representations (ICLR)*.

Christoph Lüscher, Eugen Beck, Kazuki Irie, Markus Kitza, Wilfried Michel, Albert Zeyer, Ralf Schlüter, and Hermann Ney. 2019. RWTH asr systems for librispeech: Hybrid vs attention. In *INTERSPEECH 2019 – 20th Annual Conference of the International Speech Communication Association*, pages 231–235.

Michael McCloskey and Neal J. Cohen. 1989. Catastrophic interference in connectionist networks: The sequential learning problem. 24:109–165.

Zhong Meng, Jinyu Li, and Yifan Gong. 2019. Adversarial speaker adaptation. In *2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5721–5725.

Yajie Miao, Mohammad Gowayyed, and Florian Metze. 2015. EESEN: End-to-end speech recognition using deep rnn models and wfst-based decoding. In *2015 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*, pages 167–174.

Tsubasa Ochiai, Shinji Watanabe, Shigeru Katagiri, Takaaki Hori, and John Hershey. 2018. Speaker adaptation for multichannel end-to-end speech recognition. In *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 6707–6711. IEEE.

Jia Pan, Diyuan Liu, Genshun Wan, Jun Du, Qingfeng Liu, and Zhongfu Ye. 2018. Online speaker adaptation for LVCSR based on attention mechanism. *2018 Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA ASC)*, pages 183–186.

Vassil Panayotov, Guoguo Chen, Daniel Povey, and Sanjeev Khudanpur. 2015. Librispeech: An ASR corpus based on public domain audio books. In *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE.

Daniel Povey, Arnab Ghoshal, Gilles Boulianne, Lukas Burget, Ondrej Glembek, Nagendra Goel, Mirko Hannemann, Petr Motlicek, Yanmin Qian, Petr Schwarz, Jan Silovsky, Georg Stemmer, and Karel Vesely. 2011. The kaldi speech recognition toolkit. In *2011 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*.

Lahiru Samarakoon and Khe Chai Sim. 2016a. Factorized hidden layer adaptation for deep neural network based acoustic modeling. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 24(12):2241–2250.

Lahiru Samarakoon and Khe Chai Sim. 2016b. Subspace LHUC for fast adaptation of deep neural network acoustic models. In *INTERSPEECH 2016 – 17th Annual Conference of the International Speech Communication Association*, pages 1593–1597.

George Saon, Hagen Soltau, David Nahamoo, and Michael Picheny. 2013. Speaker adaptation of neural network acoustic models using i-vectors. In *2013 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*.

Frank Seide, Gang Li, Xie Chen, and Dong Yu. 2011. Feature engineering in context-dependent deep neural networks for conversational speech transcription. In *2011 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*.

Andrew Senior and Ignacio Lopez-Moreno. 2014. Improving DNN speaker independence with i-vector inputs. In *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*.

Sabato Marco Siniscalchi, Jinyu Li, and Chin-Hui Lee. 2013. Hermitian polynomial for speaker adaptation of connectionist speech recognition systems. *IEEE Transactions on Audio, Speech, and Language Processing*, 21(10):2152–2161.

Pawel Swietojanski, Jinyu Li, and Steve Renals. 2016. Learning hidden unit contributions for unsupervised acoustic model adaptation. *IEEE/ACM Transactions on Audio, Speech and Language Processing*, 24(8):1450–1463.

Natalia Tomashenko and Yannick Estève. 2018. Evaluation of feature-space speaker adaptation for end-to-end acoustic models. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*, pages 5998–6008.

Karel Vesely, Shinji Watanabe, Katerina Zmolikova, Martin Karafiat, Lukas Burget, and Jan Honza Cernocky. 2016. Sequence summarizing neural network for speaker adaptation. In *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5315–5319. IEEE.

Shinji Watanabe, Takaaki Hori, Shigeki Karita, Tomoki Hayashi, Jiro Nishitoba, Yuya Unno, Nelson Enrique Yalta Soplin, Jahn Heymann, Matthew Wiesner, Nanxin Chen, Adithya Renduchintala, and Tsubasa Ochiai. 2018. Espnet: End-to-end speech processing toolkit. In *INTERSPEECH 2018 – 19th Annual Conference of the International Speech Communication Association*, pages 2207–2211.

Xurong Xie, Xunying Liu, Tan Lee, Shoukang Hu, and Lan Wang. 2019. BLHUC: Bayesian learning of hidden unit contributions for deep neural network speaker adaptation. In *2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5711–5715.

Kaisheng Yao, Dong Yu, Frank Seide, Hang Su, Li Deng, and Yifan Gong. 2012. Adaptation of context-dependent deep neural networks for automatic speech recognition. In *2012 IEEE Spoken Language Technology Workshop (SLT)*. IEEE.

Zhao You, Dan Su, Jie Chen, Chao Weng, and Dong Yu. 2019. DFSMN-SAN with persistent memory model for automatic speech recognition. *arXiv preprint arXiv:1910.13282*.

Dong Yu, Kaisheng Yao, Hang Su, Gang Li, and Frank Seide. 2013. KL-divergence regularized deep neural network adaptation for improved large vocabulary speech recognition. In *2013 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 7893–7897.

Kai Yu and Mark J. F. Gales. 2006. Discriminative cluster adaptive training. *IEEE Transactions on Audio, Speech, and Language Processing*, 14(5):1694–1703.

Yu Zhang, William Chan, and Navdeep Jaitly. 2017. Very deep convolutional networks for end-to-end speech recognition. In *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 4845–4849. IEEE.

Yingzhu Zhao, Chongjia Ni, Cheung-Chi Leung, Shafiq Joty, Eng Siong Chng, and Bin Ma. 2020. Speech transformer with speaker aware persistent memory. In *INTERSPEECH 2020 – 21st Annual Conference of the International Speech Communication Association*, pages 1261–1265.

Michael Zhu and Suyog Gupta. 2018. To prune, or not to prune: exploring the efficacy of pruning for model compression. In *2018 International Conference on Learning Representations (ICLR)*.

A Appendix

A.1 Details of Librispeech dataset

We use open-source Librispeech dataset for all our experiments, which is downloadable from `https://www.openslr.org/12`. Table 5 shows the statistics of the dataset.

LibriSpeech	
Training set	100h (251 speakers)
Dev_clean set	5.4h (20 males, 20 females)
Dev_other set	5.3h (17 males, 16 females)
Test_clean set	5.4h (20 males, 20 females)
Test_other set	5.1h (16 males, 17 females)

Table 5: Statistics of Librispeech dataset used for experiments.

Algorithm	**Training**	**Adaptation**
Baseline	2d13h	-
Finetune	2d13h	2min
I-vec	2d12h	2min
Pruning	2d5h	2min
Pruning+I-vec	2d5h	2min

Table 6: Average runtime.

A.2 Average Runtime

In Table 6, we list the average runtime using one V100 GPU of 1) Baseline, 2) Finetune: directly finetuning the trained baseline model, 3) I-vec: speaker-aware persistent memory method by adding i-vectors, 4) Pruning: gradual pruning, 5) Pruning+I-vec: combining feature adaptation and model adaptation methods proposed. During training, all models are trained with the given 100h Librispeech training data, while during adaptation, all models are trained with 10 utterances of adaptation data, except for the baseline where no adaptation is performed.

A.3 Evaluation Metrics

We evaluate model performance by word error rate (WER), which can be computed as following:

$$WER = \frac{S + D + I}{N_r} = \frac{S + D + I}{S + D + C} \quad (14)$$

where S is the number of substitutions, D is number of deletions, I is the number of insertions, N_r is number of words in the reference ($N_r = S + D + C$), C is the number of correct words.

A.4 Computing Infrastructure

We conduct our experiments on NVIDIA V100 GPU and Intel(R) Xeon(R) Platinum 8163 32-core CPU @ 2.50GHz.