Edinburgh Research Explorer

Speaker adaptive training using model agnostic meta-learning

OPEN ACCESS

SPEAKER ADAPTIVE TRAINING USING MODEL AGNOSTIC META-LEARNING

Ondřej Klejch, Joachim Fainberg, Peter Bell, Steve Renals

Centre for Speech Technology Research, University of Edinburgh, United Kingdom

ABSTRACT

Speaker adaptive training (SAT) of neural network acoustic models learns models in a way that makes them more suitable for adaptation to test conditions. Conventionally, model-based speaker adaptive training is performed by having a set of speaker dependent parameters that are jointly optimised with speaker independent parameters in order to remove speaker variation. However, this does not scale well if all neural network weights are to be adapted to the speaker. In this paper we formulate speaker adaptive training as a meta-learning task, in which an adaptation process using gradient descent is encoded directly into the training of the model. We compare our approach with test-only adaptation of a standard baseline model and a SAT-LHUC model with a learned speaker adaptation schedule and demonstrate that the meta-learning approach achieves comparable results.

Index Terms— speaker adaptation, speaker adaptive training, model-agnostic meta-learning

1. INTRODUCTION

Adapting an acoustic model to unseen test speakers can improve the accuracy of an automatic speech recognition (ASR) system by reducing the mismatch between training and test conditions. Speaker adaptive training (SAT) [1] further reduces the mismatch by removing speaker variance during training of the acoustic model, thus allowing it to focus solely on modelling phonetic variations. Speaker adaptation approaches can be divided into three groups: *Feature-space* methods estimate a transformation of the acoustic features to improve performance on the test data [2, 3]; *Model-based* methods adapt the weights of an acoustic model [4, 5, 6, 7, 8]; and *Hybrid* methods use auxiliary features to inform the acoustic model about speaker identity [9, 10]. These approaches are complementary and can be used together [11].

Speaker adaptive training may be applied to all these adaptation approaches. In feature-space adaptation, training of the acoustic model weights is interleaved with estimation of speaker-dependent feature transformations in order to remove speaker variability and to focus the acoustic model

on phonetic variations [1]. In model-based speaker adaptive training, the acoustic model is parameterised as speaker-dependent and speaker-independent weights. A copy of the speaker-dependent weights is maintained and optimised separately for each speaker during the training process in order to factor out speaker variation from the canonical speaker-independent acoustic model [12, 13, 14]. Finally, all hybrid approaches can be considered as speaker adaptive training because they provide information about speaker identity, which allows the acoustic model to easily remove speaker variation from the input features [9, 10, 15]. In this paper we focus on model-based adaptation and speaker adaptive training for model-based adaptation.

The biggest challenge of speaker adaptation is to improve performance on test data as much as possible without overfitting to test transcripts. This is especially important in a rapid adaptation setting when we use only a small amount of adaptation data to adapt the weights of an acoustic model. Therefore, model-based speaker adaptation usually adapts only a small number of speaker-dependent weights of an acoustic model in order not to overfit to the adaptation data. Traditionally, activations of hidden layers are adapted using linear transformations. Depending on whether we adapt inputs, hidden activations or outputs we talk about Linear Input Network (LIN), Linear Hidden Network (LHN) or Linear Output Network (LON) respectively [16, 17]. Even then these linear transformations might use too many weights, therefore low-rank approximations may be used previously [5, 18]. Learning Hidden Unit Contributions (LHUC) [4, 19] is a special case of such a transformation, in which we force the speaker dependent matrix to be diagonal, thus learning a magnitude for each hidden unit. Another similar approach adapts the scale and offset of the batch normalisation layers [8].

The methods described above require considerable manual work to carefully select which weights should be treated as speaker dependent. Moreover, these methods are not able to perform speaker adaptive training of all weights of the acoustic model which limits the expressivity of the speaker dependent transformations. This is because it is not possible to store all weights of the acoustic model for each speaker in memory and we usually do not have enough speaker dependent training data to reliably train all those weights. Furthermore, even though the acoustic model was trained in a speaker adaptive way using these methods, we still need to

find a reliable adaptation schedule to perform well in speaker adaptation. Although these problems might be solved with a careful hyper-parameter search, we can address them more explicitly by formulating the problem as a *meta-learning* task.

The aim of meta-learning is to replace handcrafted algorithms with learned algorithms, similar to how deep learning replaced handcrafted features in traditional machine learning. Recently, meta-learning has been successfully used for learning task specific learning schedules that produce better results than traditional learning schedules [20], learning good initialisations for few-shot classifiers [21, 22] or language transfer in low-resource neural machine translation [23].

Previously speaker adaptation was formulated as a meta-learning task [24]. In this paper we extend that approach to perform speaker adaptive training using a method called Model Agnostic Meta-Learning (MAML) [22]. This is a new perspective on speaker adaptive training because rather than using speaker dependent weights to remove speaker variation from the data, we are directly encoding the gradient-descent based speaker adaptation process into the training of the acoustic model. Our objective is to find weights more amenable to speaker adaptation by steering the training of the model to find regions that are better suited to rapid adaptation.

The contributions of this paper are:

1. We experimentally show that a baseline state-of-the-art acoustic model adapted with a schedule learned with MAML achieves significantly better results than the handcrafted adaptation schedule used in [24].

2. We explain how to adapt models that use batch normalisation [25], and show that it is better to use global statistics rather than batch statistics during adaptation.

3. We show that speaker adaptive training using MAML achieves comparable results with the baseline approaches.

2. ADAPTATION AS META-LEARNING

Speaker adaptation has previously been formulated as a meta-learning task [24]. In this section we review this formulation and show how it can be extended to speaker adaptive training.

2.1. Speaker Adaptation as a Meta-Learning Task

The goal of speaker adaptation is to adapt an acoustic model $f(x; \Theta)$ with weights Θ to perform better on a speaker a using adaptation data $D_a = (X_a, Y_a)$. Speaker adaptation may be formulated as a function that given an acoustic model $f(x; \Theta)$ and adaptation data D_a produces adapted weights Θ':

$$adapt(f, \Theta, D_a) \to \Theta' . \tag{1}$$

Depending on the scenario, the labels Y_a, corresponding to the acoustic input X_a, might be obtained from a reference

transcript (supervised adaptation) or obtained from the best path of a first pass decode (unsupervised adaptation).

The performance of an acoustic model on unseen data $D_u = (X_u, Y_u)$ is measured as the value of a loss function:

$$L(Y_u, f(X_u; \Theta)), \tag{2}$$

for example categorical cross-entropy, frame error rate or word error rate (WER). Note that the labels Y_u are always obtained from the reference transcripts since we want to measure the true performance of the model. Similarly, we measure the loss of an adapted acoustic model by:

$$L(Y_u, f(X_u; adapt(f, \Theta, D_a))). \tag{3}$$

To train the adaptation function using the meta-learning approach, we require the function to be both parametric and differentiable. We therefore add parameters Φ to the adaptation function (Our choice of Φ will be defined in Section 2.3.):

$$adapt(f, \Theta, D_a; \Phi) \to \Theta' . \tag{4}$$

We are now ready to introduce the loss of the meta-learner. Recall that the goal of speaker adaptation is to adapt an acoustic model $f(x; \Theta)$ using adaptation data D_a in order to improve performance on unseen data D_u. The loss of the meta-learner can then be expressed as a sum of losses of the adapted models (Equation 3) over a meta set $\mathcal{D}$:

$$J = \sum_{\mathcal{D}} L(Y_u, f(X_u; adapt(f, \Theta, D_a; \Phi))), \tag{5}$$

where $\mathcal{D}$ consists of tuples of adaptation data for the adaptation of the acoustic model, and unseen data for the evaluation of the adapted acoustic model:

$$\mathcal{D} = \{(D_{a_1}, D_{u_1}), \ldots, (D_{a_N}, D_{u_N})\} . \tag{6}$$

In theory, we should use an unlimited amount of unseen data for the evaluation of the adaptation algorithm. However, this is not practical when training the meta-learner. We approximate it by using n seconds of speech as adaptation data and the following n seconds of speech as unseen data. Further, as in any other machine learning problem we split the meta set $\mathcal{D}$ into a meta-training set $\mathcal{D}_{train}$ and a meta-validation set $\mathcal{D}_{val}$. The sets are split such that they contain different speakers, so that we can correctly assess the generalisation of the adaptation function to unseen speakers.

Finally, we use the loss J to optimise the parameters Φ of the adaptation function using gradient descent:

$$\hat{\Phi} = \arg \min_{\Phi} J . \tag{7}$$

2.2. Speaker Adaptive Training as a Meta-Learning Task

Speaker adaptive training is traditionally used to factor out speaker variation in order to enable a canonical model to focus on modelling phonological variances [1]. In model-based

speaker adaptive training this may be done by splitting the weights of the acoustic model into a speaker-independent and a speaker-dependent set. During training, a copy of the speaker-dependent weights is maintained and optimised for each speaker separately [12, 13, 14]. Here, we take an alternative approach: Instead of maintaining and optimising a separate copy of speaker-dependent weights for each speaker we embed speaker adaptation directly into the acoustic model training using a meta-learning approach in order to find a good initialisation for speaker-dependent weights. The motivation for this is that the learned initialisation of the speaker-dependent weights ought to be more amenable to speaker adaptation compared to weights obtained through standard acoustic model training. Moreover, since this approach does not have to maintain a copy of speaker dependent weights for each speaker, it is possible to train all weights in a speaker adaptive fashion and allow the model to work out which weights should be adapted.

We described how to train a meta-learner for speaker adaptation above. To formulate speaker adaptive training as a meta-learning task, we jointly optimise the weights of the acoustic model Θ and the parameters of the meta-learner Φ, minimizing the loss J (Equation 5):

$$\hat{\Theta}, \hat{\Phi} = \arg\min_{\Theta, \Phi} J \qquad (8)$$

Speaker adaptive training using the meta-learning approach (SAT-MAML) is outlined in Algorithm 1.

2.3. Implementation of the adaptation function

In order to train the adaptation function with Equation (8), the adaptation function must be differentiable. Previously [24], the adaptation function was implemented as a coordinate-wise meta-learner [20, 21] that uses a long short-term memory (LSTM) model [26] to predict adapted weights for each weight $\theta \in \Theta$ of the acoustic model. More specifically, the LSTM predicts a forget gate f and input gate i that are used in an update rule for weight θ at time-step $t + 1$:

$$\theta_{t+1} = f\theta_t - i\nabla_{\theta_t}L(Y_a, f(X_a; \Theta_t)) \qquad (9)$$

The LSTM uses three values as its inputs: the current weight value θ_t, the current loss value $L(Y_a, f(X_a; \Theta_t))$ and the corresponding gradient $\nabla_{\theta_t}L(Y_a, f(X_a; \Theta_t))$. By using these three inputs the LSTM has expressive power to learn an adaptation schedule, and also to learn to escape local minima by resetting individual weights if the current loss is high and the corresponding gradients are small [21]. In order to adapt all weights Θ of the acoustic model a large batch with dimension $|\Theta| \times 3$ for each time-step is formed as an input to the LSTM. Unfortunately, training of the coordinate-wise meta-learner quickly becomes very memory inefficient for anything but a small number of speaker dependent weights because we have to store hidden states of the LSTM for all speaker dependent weights.

Algorithm 1 SAT-MAML

Require: Number of iterations
Require: Number of speakers per batch
Require: Number of adaptation steps
Require: Learning rate α
1: **function** SAT-MAML($f, \mathcal{D}$)
2:     $\Theta \leftarrow$ random initialization
3:     $\Phi \leftarrow 0.001$
4:     **for** $i \in \{1 \cdots iterations\}$ **do**
5:         **for** $s \in \{1 \cdots speakers\_per\_batch\}$ **do**
6:             $D_a, D_u \leftarrow$ sample($\mathcal{D}$)
7:             $J_s \leftarrow L(Y_u, f(X_u; \text{ADAPT}(f, \Theta, D_a; \Phi)))$
8:
9:         $J \leftarrow \sum J_s$
10:         $\Theta \leftarrow \Theta - \alpha\nabla_{\Theta}J$
11:         $\Phi \leftarrow \Phi - \alpha\nabla_{\Phi}J$
12:
13: **return** Θ, Φ

Algorithm 2 Adaptation function

1: **function** ADAPT($f, \Theta, D_a; \Phi$)
2:     $\Theta_0 \leftarrow \Theta$
3:     **for** $j \in \{1 \cdots adaptation\_steps\}$ **do**
4:         $L_j \leftarrow L(Y_a, f(X_a; \Theta_{j-1}))$
5:         $\Theta_j \leftarrow \Theta_{j-1} - \Phi\nabla_{\Theta_{j-1}}L_j$
6: **return** $\Theta_{adaptation\_steps}$

Given the computational complexity of the coordinate-wise meta-learner, we implemented the adaptation function as in model-agnostic meta-learning (MAML) [22]. This is more memory- and computationally-efficient, yet it has the same modelling power as the coordinate-wise meta-learner [27]. With MAML the adaptation function is implemented as n steps of gradient descent, where the set of learnable parameters Φ consists only of the learning rate α, such that $\Phi = \{\alpha\}$. A single step of the adaptation function (Algorithm 2) is then implemented as:

$$adapt(f, \Theta, D_a; \{\alpha\}) = \Theta - \alpha\nabla_{\Theta}L(Y_a, f(X_a; \Theta)). \quad (10)$$

Note that in the original MAML implementation [22] the learning rate α was predefined and fixed during training, but it has been shown that the learning rate α can be trained jointly with the weights Θ [28]. Also note that meta-learning approaches use second order derivatives to train the weights Θ and the parameters Φ of the adaptation function. It has, however, been demonstrated that using only the first order derivatives also works, and this is faster and more stable during training [21, 22]. Hence, we chose to use only the first order derivatives in our experiments.

2.4. Batch normalisation in MAML

Since all our baseline models use batch normalisation [25], we also wanted to use it in the MAML models. In order to explain why using batch normalisation in MAML is complicated let us first briefly describe how batch normalisation works. Batch normalisation normalises hidden activations h in the following way:

$$h' = \gamma \frac{h - \mu}{\sqrt{\sigma^2 + \epsilon}} + \beta, \qquad (11)$$

where γ and β are the learned scale and offset weights, μ and σ^2 are the mean and variance statistics estimated on the current batch during training (denoted μ_B and σ_B^2) or as running statistics during inference (denoted μ_G and σ_G^2), and ϵ is a small number preventing division by 0. Previous papers used batch normalisation with mean and variance statistics computed only on the current batch [22] or accumulated different running statistics for each training step, because there was a big shift in distributions of hidden activations between different training steps [28].

We believe that using statistics computed only on the current batch is not optimal, because it forces the model to perform batch normalisation per utterance during inference. And since each utterance might have different duration, we would be using inconsistent estimates of true mean and variance for each utterance – it might be compared to performing cepstral mean and variance normalisation per utterance instead of cepstral mean and variance per speaker. This also hurts performance of the speaker adaptation because we are adapting parameters with different statistics than statistics that will be used during inference. Therefore we decided to use batch renormalisation [29]. Batch renormalisation normalises hidden activations h in the following way:

$$h' = \gamma \left(r \frac{h - \mu_B}{\sqrt{\sigma_B^2 + \epsilon}} + d \right) + \beta, \qquad (12)$$

where r and d are computed during forward pass as follows:

$$r = \frac{\sqrt{\sigma_B^2 + \epsilon}}{\sqrt{\sigma_G^2 + \epsilon}}, \qquad (13)$$

$$d = \frac{\mu_B - \mu_G}{\sqrt{\sigma_G^2 + \epsilon}} \qquad (14)$$

and are treated as constants during backward pass. This allowed us to accumulate running statistics μ_G, σ_G^2 during training and adapt the model with respect to the same hidden activations during adaptation and inference, because by substituting Equations 13 and 14 into Equation 12 we obtain:

$$h' = \gamma \frac{h - \mu_G}{\sqrt{\sigma_G^2 + \epsilon}} + \beta. \qquad (15)$$

3. EXPERIMENTS

We performed experiments on TED talks because they provide many single speaker recordings, which allowed us to train models in a speaker adaptive way. The models were trained using the TED-LIUM dataset [30] and evaluated on TED talks that were used for the IWSLT 2010, 2011 and 2012 evaluations [31, 32, 33]. To comply with the evaluation protocol, we used only TED talks that had been recorded before the end of 2012 for model training. This resulted in 130 hours of raw training data. We increased the size of the training data by a factor of 3 by performing speed perturbation [34]. The development (dev) set consists of 18 speakers with an average speech duration of 6.5 minutes and the test set consists of 30 speakers with an average speech duration of 10.8 minutes. The training scripts were implemented in Tensorflow [35] and Keras [36] and we used Kaldi [37] for decoding. The scripts are publicly available.[1]

3.1. Speaker adaptation

As a baseline we performed speaker adaptation of LHUC weights and ALL weights of a baseline model. In both methods we used 3 steps of full-batch gradient descent to adapt the weights. In all experiments we used the dev set to train a meta-learner to find the per-layer learning rate for 10s, 30s and 60s of adaptation data. We performed both supervised and unsupervised speaker adaptation experiments. Unsupervised labels were obtained using a separately trained baseline model and were used for unsupervised speaker adaptation of all models. During adaptation, we removed frames corresponding to silence from the adaptation data, because silence does not contain any speaker information.

3.2. Baseline model

All models used LDA projected [38] 40 dimensional MFCC features without cepstral mean and variance normalisation. The model was a time-delay neural network (TDNN) [39] and it consisted of 6 hidden layers with 600 neurons and a ReLU activation function [40] followed by batch normalisation [25] and an LHUC layer [4]. LHUC layers were used only for adaptation. The model predicted posteriors of 4208 tied context dependent phones. The model had 5.9M parameters in total.[2] We trained the model for 400 iterations with Adam [41]. In each iteration we trained on 2000 batches that contained 256 chunks with 8 frames. This roughly corresponds to doing 3 epochs of training on all available training data. We performed early-stopping [42, 43] by monitoring loss on a validation set after each iteration. We used the best performing model for decoding.

---

[1] https://github.com/ondrejklejch/learning_to_adapt
[2] Note, that this model is 50% smaller than a model used in a corresponding Kaldi recipe that has 850 neurons in each hidden layer. But this smaller model has only 0.5% absolute higher WER than the larger model.

3.3. SAT-LHUC

The SAT-LHUC model had the same architecture and used the same training schedule as the baseline model except that during training we used speaker specific LHUC parameters for each speaker in order to make the model able to remove speaker variance. In order to obtain a speaker-independent model, we used speaker independent LHUC weights instead of speaker-dependent LHUC weights with probability 0.5, as suggested in the original SAT-LHUC paper [14].

3.4. MAML

We trained a model with the same architecture using the MAML approach. For training of the model we used 10s of adaptation data and the goal was to improve performance on the following 10s of unseen data using three full-batch adaptation steps. We trained different models for adaptation of LHUC parameters (MAML-LHUC) and adaptation of ALL parameters (MAML-ALL). We trained the model with a combination of a loss computed with the original unadapted model and a loss computed with the adapted model. In our early experiments we found that using only the loss computed with the adapted model lead to worse results. We initialised the model with a baseline model that was trained for 200 iterations and continued training it for another 200 iterations using the MAML objective. We used Adam [41] as an optimiser. In each iteration we trained on 1024 batches that contained data for 4 different speakers. This way all models used the same number of frames for training. At the end we fine-tuned per-layer learning rates for varying amounts of adaptation data on the dev set.

4. RESULTS

All our models use batch normalisation. In our first experiment we tested how to treat the mean μ and variance statistics σ^2 of the batch normalisation when adapting the LHUC weights of the baseline model. Our results are shown in Table 1. Traditionally, batch statistics μ_B, σ_B^2 are used during training (we call them batch stats), and running mean μ_G and variance σ_G^2 statistics are used for inference (we call them global stats). The results in Table 1 suggests that for speaker adaptation with limited amounts of adaptation data it is better to use global statistics than batch statistics. This is likely because the global statistics are better estimates of the true means and variances – by adapting LHUC weights we are essentially correcting for errors in the variance estimation. (This was the only experiment where we used a learning rate of 0.7 for adaptation of LHUC weights, because it was found to work well in [24].)

In the second experiment we conducted supervised speaker adaptation of the baseline model, SAT-LHUC, and MAML models (Table 2). In all experiments we used adaptation

	global stats		batch stats	
	dev	test	dev	test
original	15.3	13.4	15.3	13.4
10s	15.1	13.1	15.2	13.3
30s	14.8	12.7	15.0	13.2
60s	14.5	12.3	14.8	13.0

Table 1. WERs (%) of supervised speaker adaptation of LHUC parameters of the baseline model. Batch normalisation statistics are estimated on all the training data (global stats), or on the adaptation data (batch stats). The global statistics are likely more robust given the small amounts of data, producing lower error rates.

schedules learned with the meta-learning approach (Equation 7), instead of a hand-crafted adaptation schedule.

When adapting all parameters, MAML-ALL typically outperforms the baseline, particularly for 60s of adaptation data. This suggests that it has found an improved schedule using MAML, compared to the hand-crafted schedule for Baseline-ALL.

In the experiments adapting the LHUC weights of the baseline model, using the learned adaptation schedule achieves much better results (the first column in Table 2) than adaptation using a handcrafted learning rate schedule (the first column in Table 1). This is because the learned learning rates are $3\times - 4\times$ larger than the handcrafted learning rate 0.7, which was found to work well in [24]. This result has a straightforward explanation: In this paper we perform full-batch adaptation, whereas in [24], adaptation was performed with a batch size of 256 frames. Consequently, adapting using a full batch of 10 s (1000 frames) is approximately $4\times$ larger than a batch size of 256. Nevertheless, this highlights the usefulness of using the meta-learning approach for estimating the adaptation schedule compared to an excessive hyper-parameter search, which requires to select appropriate step-sizes and bounds on the hyper-parameters. We argue that the learned learning rates for LHUC layers are several orders of magnitude larger than commonly used learning rates. There is therefore a considerable chance that the hyper-parameter bounds would not be set optimally to uncover the best solution. We observed similar trends when we performed unsupervised speaker adaptation experiments (Table 3), but obtained much higher WERs than with supervised speaker adaptation.

When we compare the baseline model with SAT-LHUC and MAML-LHUC models we see that there is not a big difference between adaptation of these models. There are several possible explanations for this observation. First, SAT-LHUC was originally shown to improve speaker adaptation with a feed-forward neural network [14]. This feed-forward neural network was approximately $5\times$ bigger than our TDNN baseline, but it achieved much worse performance before adap-

	Baseline-LHUC		Baseline-ALL		SAT-LHUC		MAML-LHUC		MAML-ALL	
	dev	test	dev	test	dev	test	dev	test	dev	test
original	15.3	13.4	15.3	13.4	15.4	13.5	15.3	13.4	15.3	13.4
10s	14.8	12.7	14.7	12.6	14.9	12.6	14.8	12.7	14.7	12.5
30s	14.5	12.2	14.2	12.0	14.4	12.1	14.4	12.2	14.3	11.9
60s	14.2	12.0	14.0	11.7	14.1	11.9	14.2	11.9	13.8	11.5

Table 2. WER (%) for supervised speaker adaptation of the baseline, SAT-LHUC, MAML-LHUC, and MAML-ALL models.

	Baseline-LHUC		Baseline-ALL		SAT-LHUC		MAML-LHUC		MAML-ALL	
	dev	test	dev	test	dev	test	dev	test	dev	test
original	15.3	13.4	15.3	13.4	15.4	13.5	15.3	13.4	15.3	13.4
10s	15.1	13.0	15.1	13.0	15.2	12.9	15.1	13.0	15.0	13.0
30s	15.0	12.9	14.9	12.9	15.1	12.8	15.0	12.8	15.0	12.9
60s	14.8	12.9	14.8	12.8	14.9	12.8	14.8	12.9	14.8	12.8

Table 3. WER (%) for unsupervised speaker adaptation of the baseline, SAT-LHUC, MAML-LHUC, and MAML-ALL models.

tation. It is possible that our TDNN baseline is sufficiently powerful to model both speaker and phonological variability inside the canonical model. It therefore might not benefit from factoring out speaker variability into speaker dependent weights as much as the feed-forward neural network. Second, the feed-forward neural network used in [14] did not employ batch normalisation. It is possible that the batch normalisation may implicitly be removing speaker variation from the data, thus removing the need for speaker adaptation. Third, there are only 881 speakers in the training data and differences between them are probably much smaller than differences between different classes in the few-shot learning scenario [22], therefore the model is not forced to factor out speaker variability into speaker dependent weights. We plan to address all these hypotheses in our future work by running experiments with different model architectures, with and without batch normalisation, and by using data augmentation to introduce greater speaker variability into the training set.

5. CONCLUSIONS

In this paper we have extended the meta-learning approach for speaker adaptation [24] to support speaker adaptive training. The aim of speaker adaptive training is to remove speaker variability from the data in order for the canonical acoustic model to focus solely on modelling phonetic variation. Previous speaker adaptive training approaches used speaker-dependent weights to factor out speaker variability during training. Rather than removing speaker variation during training, we proposed to use meta-learning to embed speaker adaptation via gradient descent directly into the training of the acoustic model. We hypothesised that this should result in the acoustic model learning weights that are more amenable to test-time adaptation, because the optimisation process using

the meta-learning objective should steer weights of the acoustic model to regions that allow rapid adaptation. However, in our experiments we found that speaker adaptive training – using SAT-LHUC or MAML – did not improve performance of speaker adaptation of a strong state-of-the-art baseline model.

We showed that using global statistics for batch normalisation is beneficial compared to using statistics computed on the adaptation data in the low data adaptation regime, because global statistics are better estimates of the true mean and variance. We then demonstrated that using the meta-learner for estimation of an adaptation schedule achieves good results without any need for an excessive hyper-parameter search that still requires good bounds on the hyper-parameters.

In future work we plan to further analyse why speaker adaptive training of current state-of-the-art models does not yield improvements. We would also like to devise a method that will perform better in rapid adaptation and unsupervised adaptation. Finally, we plan to evaluate meta-learning on larger tasks such as domain adaptation or language adaptation of ASR models.

6. REFERENCES

[1] T. Anastasakos, J. McDonough, R. Schwartz, and J. Makhoul, "A compact model for speaker-adaptive training," in *ICSLP*, 1996.

[2] C. J. Leggetter and P. C. Woodland, "Maximum likelihood linear regression for speaker adaptation of continuous density hidden markov models," *Computer Speech & Language*, vol. 9, no. 2, pp. 171–185, 1995.

[3] M. J. F. Gales, "Maximum likelihood linear transformations for HMM-based speech recognition," *Computer Speech & Language*, vol. 12, no. 2, pp. 75–98, 1998.

[4] P. Swietojanski, J. Li, and S. Renals, "Learning hidden unit contributions for unsupervised acoustic model adaptation," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 14, pp. 1450–1463, 2016.

[5] J. Xue, J. Li, D. Yu, M. Seltzer, and Y. Gong, "Singular value decomposition based low-footprint speaker adaptation and personalization for deep neural network," in *ICASSP*, 2014.

[6] H. Liao, "Speaker adaptation of context dependent deep neural networks," in *ICASSP*, 2013.

[7] D. Yu, K. Yao, H. Su, G. Li, and F. Seide, "KL-divergence regularized deep neural network adaptation for improved large vocabulary speech recognition," in *ICASSP*, 2013.

[8] Z. Q. Wang and D. Wang, "Unsupervised speaker adaptation of batch normalized acoustic models for robust ASR," in *ICASSP*, 2017.

[9] G. Saon, H. Soltau, D. Nahamoo, and M. Picheny, "Speaker adaptation of neural network acoustic models using i-vectors," in *ASRU*, 2013.

[10] O. Abdel-Hamid and H. Jiang, "Fast speaker adaptation of hybrid NN/HMM model for speech recognition based on discriminative learning of speaker code," in *ICASSP*, 2013.

[11] L. Samarakoon and K. C. Sim, "On combining i-vectors and discriminative adaptation methods for unsupervised speaker normalization in DNN acoustic models," in *ICASSP*, 2016.

[12] T. Ochiai, S. Matsuda, H. Watanabe, X. Lu, Ch. Hori, and S. Katagiri, "Speaker adaptive training for deep neural networks embedding linear transformation networks," in *ICASSP*, 2015.

[13] T. Ochiai, S. Matsuda, H. Watanabe, X. Lu, H. Kawai, and S. Katagiri, "Bottleneck linear transformation network adaptation for speaker adaptive training-based hybrid DNN-HMM speech recognizer," in *ICASSP*, 2016.

[14] P. Swietojanski and S. Renals, "SAT-LHUC: Speaker adaptive training for learning hidden unit contributions," in *ICASSP*, 2016.

[15] Y. Miao, H. Zhang, and F. Metze, "Speaker adaptive training of deep neural network acoustic models using i-vectors," *IEEE/ACM Transactions on Audio, Speech and Language Processing (TASLP)*, vol. 23, no. 11, pp. 1938–1949, 2015.

[16] R. Gemello, F. Mana, S. Scanzio, P. Laface, and R. De Mori, "Adaptation of hybrid ANN/HMM models using linear hidden transformations and conservative training," in *ICASSP*, 2006.

[17] R. Gemello, F. Mana, S. Scanzio, P. Laface, and R. De Mori, "Linear hidden transformations for adaptation of hybrid ANN/HMM models," *Speech Communication*, vol. 49, no. 10-11, pp. 827–835, 2007.

[18] Y. Zhao, J. Li, K. Kumar, and Y. Gong, "Extended low-rank plus diagonal adaptation for deep and recurrent neural networks," in *ICASSP*, 2017.

[19] P. Swietojanski and S. Renals, "Learning hidden unit contributions for unsupervised speaker adaptation of neural network acoustic models," in *SLT*, 2014.

[20] M. Andrychowicz, M. Denil, S. Gomez, M. W. Hoffman, D. Pfau, T. Schaul, and N. de Freitas, "Learning to learn by gradient descent by gradient descent," in *NIPS*, 2016.

[21] S. Ravi and H. Larochelle, "Optimization as a model for few-shot learning," in *ICLR*, 2017.

[22] Ch. Finn, P. Abbeel, and S. Levine, "Model-agnostic meta-learning for fast adaptation of deep networks," in *ICML*, 2017.

[23] J. Gu, Y. Wang, Y. Chen, K. Cho, and V. O. K. Li, "Meta-learning for low-resource neural machine translation," in *ACL*, 2018.

[24] O. Klejch, J. Fainberg, and P. Bell, "Learning to adapt: a meta-learning approach for speaker adaptation," in *Interspeech*, 2018.

[25] S. Ioffe and Ch. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *ICML*, 2015.

[26] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[27] Ch. Finn and S. Levine, "Meta-learning and universality: Deep representations and gradient descent can approximate any learning algorithm," in *ICLR*, 2018.

[28] A. Antoniou, H. Edwards, and A. Storkey, "How to train your MAML," in *ICLR*, 2019.

[29] S. Ioffe, "Batch renormalization: Towards reducing minibatch dependence in batch-normalized models," in *NeurIPS*, 2017.

[30] A. Rousseau, P. Deléglise, and Y. Estève, "TED-LIUM: an automatic speech recognition dedicated corpus," in *LREC*, 2012.

[31] M. Paul, M. Federico, and S. Stücker, "Overview of the IWSLT 2010 evaluation campaign," in *IWSLT*, 2010.

[32] M. Federico, L. Bentivogli, M. Paul, and S. Stücker, "Overview of the IWSLT 2011 evaluation campaign," in *IWSLT*, 2011.

[33] M. Federico, M. Cettolo, L. Bentivogli, M. Paul, and S. Stüker, "Overview of the IWSLT 2012 evaluation campaign," in *IWSLT*, 2012.

[34] T. Ko, V. Peddinti, D. Povey, and S. Khudanpur, "Audio augmentation for speech recognition," in *Interspeech*, 2015.

[35] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard, et al., "Tensorflow: A system for large-scale machine learning.," in *OSDI*, 2016, vol. 16, pp. 265–283.

[36] F. Chollet et al., "Keras," `https://github.com/keras-team/keras`, 2015.

[37] D. Povey, A. Ghoshal, G. Boulianne, L. Burget, O. Glembek, N. Goel, M. Hannemann, P. Motlíček, Y. Qian, P. Schwarz, J. Silovský, G. Stemmer, and K. Veselý, "The Kaldi speech recognition toolkit," in *ASRU*, 2011.

[38] R. O. Duda, P. E. Hart, and D. G. Stork, *Pattern classification*, Wiley, 2012.

[39] V. Peddinti, D. Povey, and S. Khudanpur, "A time delay neural network architecture for efficient modeling of long temporal contexts.," in *Interspeech*, 2015.

[40] V. Nair and G. E. Hinton, "Rectified linear units improve restricted boltzmann machines," in *ICML*, 2010.

[41] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[42] N. Morgan and H. Bourlard, "Continuous speech recognition using multilayer perceptrons with hidden Markov models," in *ICASSP*, 1990, pp. 413–416.

[43] L. Prechelt, "Automatic early stopping using cross validation: quantifying the criteria," *Neural Networks*, vol. 11, no. 4, pp. 761–767, 1998.