Interspeech 2018
2-6 September 2018, Hyderabad

Learning to adapt: a meta-learning approach for speaker adaptation

Ondřej Klejch, Joachim Fainberg, Peter Bell

Centre for Speech Technology Research, University of Edinburgh, Edinburgh EH8 9AB, UK
{o.klejch, j.fainberg, peter.bell}@ed.ac.uk

Abstract

The performance of automatic speech recognition systems can be improved by adapting an acoustic model to compensate for the mismatch between training and testing conditions, for example by adapting to unseen speakers. The success of speaker adaptation methods relies on selecting weights that are suitable for adaptation and using good adaptation schedules to update these weights in order not to overfit to the adaptation data. In this paper we investigate a principled way of adapting all the weights of the acoustic model using a meta-learning. We show that the meta-learner can learn to perform supervised and unsupervised speaker adaptation and that it outperforms a strong baseline adapting LHUC parameters when adapting a DNN AM with 1.5M parameters. We also report initial experiments on adapting TDNN AMs, where the meta-learner achieves comparable performance with LHUC.

Index Terms: automatic speech recognition, speaker adaptation, meta-learning

1. Introduction

The accuracy of automatic speech recognition (ASR) systems can be improved by adapting their acoustic models (AM) to new speakers with available adaptation data. There are three possible approaches to speaker adaptation: feature-space approaches estimate feature transforms to maximise the log-likelihood of the adaptation data [1, 2]; model-based approaches update the parameters of the AM [3, 4, 5, 6]; and hybrid approaches use auxiliary features to provide information about the speaker to the AM [7, 8].

In this paper we focus on model-based adaptation of neural network AMs. The aim is to compensate for a mismatch between training and testing data by adapting the weights of an AM. This may include adapting all the weights [5, 9] or only a subset of the weights [3, 4, 10, 11]. The biggest problem of adapting all the weights is that in scenarios with small amounts of adaptation data it tends to overfit. Methods that adapt only a subset of the weights try to overcome this problem by limiting the expressivity of the adaptation. Another problem is that adapting all the weights has a much bigger memory-footprint than adapting only a subset of the weights.

Typically methods that adapt only a subset of the weights work by inserting new weights that scale activations of some hidden layer. For example, the Learning Hidden Unit Contributions (LHUC) method [3] uses a speaker dependent vector r (called LHUC parameters in the rest of the paper) as element-wise multipliers to scale the activations of a hidden layer h:

$$h' = r \circ h. \tag{1}$$

Other methods use a speaker dependent matrix A to adapt the activations of a hidden layer h [12]:

$$h' = Ah. \tag{2}$$

Since the matrix A can be quite big in terms of the number of speaker dependent parameters that have to be estimated using adaptation data, it is usually inserted after a layer with small dimensions. That can be a bottleneck layer or it can be inserted after a bottleneck obtained by decomposing a layer into a product of two low-rank matrices using singular value decomposition (SVD) [4, 13].

Even adapting only a subset of the weights of a context-dependent AM might result in data sparsity issues arising from small amounts of adaptation data, as the posterior probabilities of senones that are unseen in the adaptation data are pushed towards zero [14]. Different regularisation techniques have been proposed to alleviate this issue, such as L2 normalisation of the difference between the original and the adapted weights [5], Kullback-Leibler divergence of the posteriors of the original and the adapted model [6] or maximum a posteriori (MAP) adaptation [14]. Alternatively, it has also been proposed to use a lower entropy task for adaptation, leveraging context-independent targets [15, 16].

While adapting only a subset of all weights benefits from a small memory footprint and a robustness to overfitting to adaptation data [3], it lacks the expressivity of adapting all weights. Therefore, we decided to find a reliable way of adapting all the weights of the AM which would eventually enable online and iterative adaptation [13]. To reliably adapt all the weights it is important to use a good adaptation schedule (number of adaptation steps, learning rate, etc.). Recently it has been shown that meta-learning can find good update rules for training neural networks [17, 18], we therefore decided to evaluate it in the context of speaker adaptation.

Meta-learning tries to replace hand-crafted algorithms with learned, specialised algorithms in a similar way to how deep learning replaced hand-crafted features with features extracted using neural networks. Recently, Andrychowicz et al. [17] used a meta-learning approach to learn task specific update rules that outperform general update rules such as Adam [19]. They did so by training a coordinate-wise meta-learner that updates individual parameters of the trained model (see Section 2.1 for more details). The same approach was later used by Ravi and Larochelle [18] to train a meta-learner for a few-shot learning scenario, where the meta-learner is required to train a new classifier for a set of unseen classes given only a few examples of each class (typically 1 or 5 examples per class). Speaker adaptation might be viewed as a special case of few-shot learning because we try to adapt a speaker-independent model to unseen speaker conditions using limited adaptation data. Therefore, we decided to evaluate the same meta-learning approach using the coordinate-wise meta-learner for speaker adaptation. There are several reasons for using meta-learning for speaker adaptation:

1. Speaker adaptation is performed many times, hence it is important to find a reliable adaptation schedule that will work for different speakers in different environments.
2. The meta-learner learns how to adapt weights, but it

ought to implicitly learn which sets of weights are suitable for adaptation in a given scenario.

3. The meta-learner ought to find the best adaptation schedule given some external constraints such as the type of the AM, the number of adaptation steps, the amount of adaptation data, supervised/unsupervised targets, etc.

In this paper we investigate the use of a coordinate-wise meta-learner [17, 18] for supervised and unsupervised speaker adaptation and we compare it with adapting only LHUC parameters [3] and adapting all the weights of the AM. The rest of the paper is organised as follows. We formulate speaker adaptation as a meta-learning task and we describe the coordinate-wise meta-learner in Section 2, we describe adaptation experiments conducted in the domain of TED talks in Section 3 and we conclude the paper in Section 4.

2. Method

As stated in Section 1, meta-learning aims to replace hand-crafted algorithms with learned task-specific algorithms. In order to train a meta-learner for a particular task it is necessary to find an appropriate meta-learning architecture and a loss function. In this section we describe these components for the meta-learner that we used for speaker adaptation.

2.1. Speaker adaptation as a meta-learning task

In this paper we view speaker adaptation as a function, $adapt$, with a set of parameters Φ that adapts a set of weights Θ of an acoustic model $f(x;\Theta)$ to a set of adapted weights Θ' using adaptation data $D = (X, Y)$, where X are acoustic features and Y are labels. It can be formalised by the equation

$$adapt(f, \Theta, D; \Phi) \rightarrow \Theta'. \tag{3}$$

Depending on the adaptation scenario, the labels Y might correspond to true labels (supervised speaker adaptation) or labels obtained from a one-best path from a first pass decoding (unsupervised speaker adaptation).

The meta-learner is then trained to adapt the weights of the original model using adaptation data such that it improves cross-entropy loss L on test data. In this paper we approximate this by splitting the training data into C equally sized chunks and training the meta-learner to adapt on chunk c containing N frames with corresponding data $D_c = (X_c, Y_c)$ to improve cross-entropy L on the following chunk $c+1$ with data $D_{c+1} = (X_{c+1}, Y_{c+1})$. The loss J for training of the meta-learner is

$$J(\Phi) = \sum_{c=1}^{C} \sum_{(x,y)\in D_{c+1}} L(y, f(x; adapt(f, \Theta, D_c, \Phi))). \tag{4}$$

We do not need access to the test data for training the meta-learner. We can train the meta-learner with held-out data that was not used for training the AM, although it is beneficial if the held-out data resembles the test data as the meta-learner can learn the most suitable adaptation schedule for the AM for the given test set.

During evaluation the meta-learner uses adaptation data $D = (X, Y)$ to improve performance on test data by predicting a set of updated weights Θ' that are then used for decoding:

$$\Theta' = adapt(f, \Theta, D; \Phi). \tag{5}$$

It is important to note that in both the supervised and the unsupervised adaptation settings we use the true labels as y_{test} for training the meta-learner, because we want to maximise performance on true labels. For y_{adapt}, however, we only use the true labels in the supervised speaker adaptation setting. In unsupervised speaker adaptation we use labels obtained from the first pass decoding.

2.2. Coordinate-wise meta-learner

The adaptation function, $adapt(f, \Theta, D; \Phi) \rightarrow \Theta'$, is implemented as a two layer LSTM with a set of parameters Φ as in [18]. The coordinate-wise meta-learner updates each weight $\theta \in \Theta$ individually – each weight θ is presented as a single data sample to the meta-learner. Note, that in practice we batch all the weights Θ and we adapt them jointly. This has two advantages. First, the parameters of the meta-learner, Φ, are shared across all weights Θ of the acoustic model. Second, the parameters of the meta-learner, Φ, have much smaller dimensionality because they do not need to work with big inputs. In the following section we will describe how the meta-learner adapts a single weight $\theta \in \Theta$ using adaptation data D. The process is also illustrated in Figure 1.

The first layer is a standard LSTM which at time step t accepts a vector v_t with three values: the current value of the weight θ_t, the current loss value L_t, and the corresponding gradient $\nabla_{\theta_t} L_t$:

$$v_t = \begin{pmatrix} \theta_t \\ L_t \\ \nabla_{\theta_t} L_t \end{pmatrix}. \tag{6}$$

The loss L_t is computed using adaptation data D and current weights Θ_t:

$$L_t = \sum_{(x,y)\in D} L(y, f(x; \Theta_t)), \tag{7}$$

and we initialise $\theta_1 = \theta$. Using this input vector v_t the first LSTM layer produces a hidden representation

$$h_t = LSTM(v_t). \tag{8}$$

The second LSTM layer uses this hidden representation h_t to predict the value of a forget gate f_t with parameters W_F and b_F:

$$f_t = \sigma(W_F \cdot [h_t, f_{t-1}] + b_F), \tag{9}$$

and to predict the value of an input gate i_t with parameters W_I and b_I:

$$i_t = \sigma(W_I \cdot [h_t, i_t] + b_I). \tag{10}$$

Both the forget gate f_t and the input gate i_t are used to update the weight θ_t to θ_{t+1} using the corresponding gradient $\nabla_{\theta_t} L$:

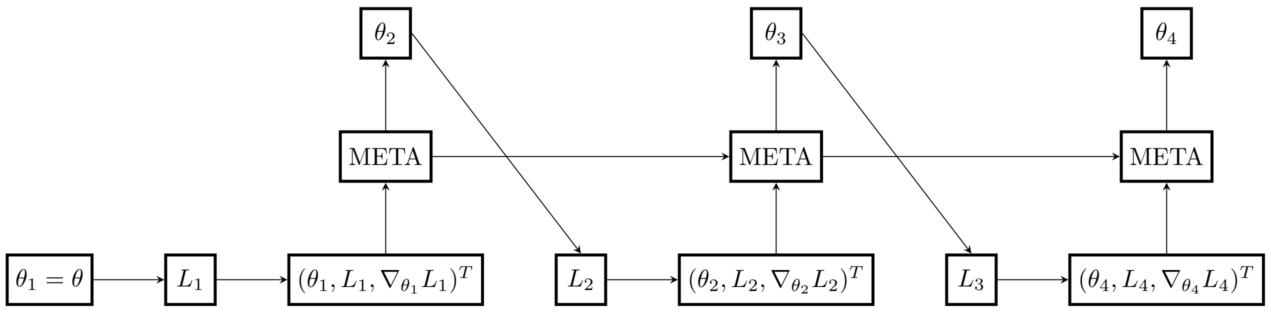
$$\theta_{t+1} = f_t \cdot \theta_t - i_t \cdot \nabla_{\theta_t} L. \tag{11}$$

In this update rule the input gate acts as a learning rate and the forget gate can be used to escape local minima when the loss is high but the gradient is close to zero [18].

Note that we followed [17] and preprocessed the losses L_t with the following method using the suggested value $p = 10$:

$$L_t \rightarrow \begin{cases} (\frac{log(|L_t|)}{p}, sgn(x)) & \text{if } |L_t| \geq e^p \\ (-1, e^p L_t), & \text{otherwise.} \end{cases} \tag{12}$$

Similar preprocessing was applied to the gradients $\nabla_\theta L_t$.



θ_2
θ_3
θ_4
META
META
META
$\theta_1 = \theta$
L_1
$(\theta_1, L_1, \nabla_{\theta_1} L_1)^T$
L_2
$(\theta_2, L_2, \nabla_{\theta_2} L_2)^T$
L_3
$(\theta_4, L_4, \nabla_{\theta_4} L_4)^T$

Figure 1: *An illustration of how the meta-learner adapts a single weight θ in three adaptation steps.*

3. Experiments

3.1. Dataset

All experiments were performed on TED talks. We used the TED-LIUM dataset [20] for training of the acoustic models and we tested these models on a combined test set of IWSLT 2010, 2011 and 2012 [21, 22, 23]. To comply with the IWSLT evaluation protocol we only used talks that were recorded before the end of 2012 for training of the acoustic model. This pruned training set contains 134 hours of training data. The combined test set contains 30 speakers and is 5.3 hours long.

3.2. Baseline DNN Setup

Training the coordinate-wise meta-learners requires large amounts of memory – linear in the number of weights of the acoustic model. Therefore, we decided to evaluate the adaptation methods on smaller acoustic models. We used Kaldi [24] to train a small deep neural network model (called DNN in Table 1 and Table 2) with 1.5M weights across 6 hidden layers, each with 256 neurons, using sigmoid activation functions and an output layer corresponding to 3792 tied context-dependent phones and an input corresponding to 7 acoustic frames.

We also experimented with the adaptation of a time-delayed neural network model (TDNN) [25] to show that the meta-learner works with state-of-the-art architectures. Again we trained a smaller model with 2.1M weights across 6 hidden layers, with 300 units each and RELU activation function and splice indexes `-2,-1,0,1,2 -1,2 -3,3 -7,2 -3,3 0`. We evaluated the adaptation of two TDNN models, one using batch-normalisation and the other using L2 normalisation[1] (TDNN-BN and TDNN-RN in Table 1 and Table 2). All models used a pruned 3-gram language model for decoding, we did not use a 4-gram language model for rescoring.

3.3. Adaptation Setup

We explored adaptation using only a small amount of adaptation data: in all the experiments we used the first 10 seconds of data to perform speaker adaptation. For the baseline adaptation experiments we either adapted all layers or only the LHUC parameters of each unit in each layer (denoted ALL and LHUC in Table 1 and Table 2). For both technique we adapted for 3 epochs using stochastic gradient descent (SGD) with a learning rate of: 0.01 for the DNN model; $2.5 \cdot 10^{-6}$ for the TDNN models (ALL); and 0.7 (LHUC) for both DNN and TDNN models.

We used the development sets from IWSLT 2010 and 2012 [21, 23] to train the meta-learner. Together these datasets contain 3 hours of audio. The data for the first 13 speakers was used as the training set and the last 5 speakers as the validation set. We trained the meta-learner to adapt the acoustic model using 10 seconds of adaptation data to improve performance on the following 10 seconds. The meta-learner was trained using Adam [19] with a learning rate of 0.001. During training we monitored the loss on the validation set. We selected the meta-learner that achieved the best validation loss for testing. Note that we initialized a bias of the input gate b_I to small values (sampled uniformly from $[-5, -4]$) and a bias of the forget gate b_F to high values (sampled uniformly from $[4, 5]$) such that the meta-learner starts training with an update rule similar to SGD.

When training a meta-learner for the DNN model we split the data into chunks of 1000 frames and we trained a meta-learner with 20 units in the hidden layer to perform one full-batch adaptation step. We also experimented with more adaptation steps, but we did not observe any improvements compared to performing a single adaptation step. When we tried the same procedure with TDNN models, the meta-learner learned to overfit to silent frames, producing too many deletion errors. This was surprising as the amount of silent frames in the alignments obtained with TDNN models was similar to those obtained by the DNN model. We hypothesized that this might be due to the much larger context (29 acoustic frames) of the TDNN models, so we therefore removed silent phones from the meta-learner training data and from the adaptation data – there is not much benefit from doing speaker adaptation on silent frames. We preprocessed the data by trimming silence at the beginning and end of each utterance, splitting each utterance into 50 ms chunks, and discarding chunks that contained more than 10% silent frames. Next we also changed the input vector to the meta-learner, v, to contain the normalised position of the weight in the weight matrix instead of the weight value θ. If the weight θ corresponded to an element at a position i, j in a weight matrix $W \in R^{m \times n}$ then the input vector v was:

$$v_t = \begin{pmatrix} i/m \\ j/n \\ L_t \\ \nabla_{\theta_t} L_t \end{pmatrix}. \tag{13}$$

This should provide the meta-learner with much finer control over which weights are suitable for adaptation instead of just learning different adaptation schedules for each layer.

We implemented the meta-learner[2] in Tensorflow [26] and Keras [27].

[1] `y = x * sqrt(dim(x)) * target-rms / |x|`

[2] `https://github.com/choko/learning_to_adapt`

	DNN	TDNN-BN	TDNN-RN
original	20.7	15.2	15.2
LHUC	20.1	14.3	14.6
ALL	20.6	14.5	14.6
META	19.8	14.4	14.6

Table 1: *WER (%) of the supervised speaker adaptation experiments using 10s of adaptation data.*

	DNN	TDNN-BN	TDNN-RN
original	20.7	15.2	15.2
LHUC	20.5	14.6	14.8
ALL	20.6	14.7	14.8
META-sup	20.0	14.7	15.0
META-unsup	19.7	14.7	14.9

Table 2: *WER (%) of the unsupervised speaker adaptation experiments using 10s of adaptation data.*

3.4. Results

In this paper we evaluated relatively small models in order to be able to train a meta-learner for them. The DNN model achieves a word error rate (WER) of 20.7% on our test set and both TDNN models achieve a WER of 15.2% which is close to a model used in [28] which achieves WER 14.9% while the TDNN models are 20× smaller.

First we performed supervised speaker adaptation using 10 seconds of adaptation data (Table 1). When adapting the LHUC parameters of the DNN model we got an improvement of 0.6% absolute which is comparable to other previous experiments [3]. Adapting all the weights performed worse as the adaptation schedule may have overfitted to the adaptation data, but the meta-learner was able to learn a good adaptation schedule that outperforms adapting LHUC parameters by another 0.3% absolute. Note that the meta-learner performed only 1 adaptation step and LHUC used 3 adaptation steps. We believe that finding a way to train the meta-learner for more adaptation steps might bring further improvements. When adapting TDNN models the meta-learners were not that successful, they were not able to match performance of LHUC (TDNN-BN) or outperform LHUC (TDNN-RN). The meta-learner may require more tuning to be able work with such complicated models as TDNNs. It is interesting to point out that the TDNN with batch normalisation benefits much more from LHUC adaptation than the TDNN trained with L2 normalisation. This finding is similar to [29] where they adapted only batch normalisation parameters during speaker adaptation.

We then performed unsupervised adaptation using 10 seconds of adaptation data (Table 2). When adapting the DNN model, the meta-learner outperforms adapting only LHUC parameters. To test whether the meta-learner is able to determine the best adaptation in the given scenario, we evaluated a meta-learner that was trained to use supervised targets for adaptation (META-sup) and a meta-learner that was trained to use unsupervised targets for adaptation (META-unsup). The results show that it is very beneficial to train the meta-learner in the same conditions as in testing. Unfortunately, unsupervised adaptation of TDNN models with a current implementation of the meta-learner did not match the performance of LHUC.

4. Conclusions

In this paper we tried to find a reliable way of adapting all the weights of a neural network AM because we believe that adapting only a subset of the weights limits expressivity of the adaptation. We investigated a meta-learning approach for finding a good adaptation schedule – we showed that speaker adaptation can be formulated as a meta-learning task by defining a loss to train a coordinate-wise meta-learner as in [17, 18] to perform supervised and unsupervised speaker adaptation using only 10 seconds of adaptation data. We compared this approach with adapting all weights of the AM and LHUC parameters using SGD. Our results on adaptation of the DNN model suggest that meta-learning might be a useful method for speaker adaptation, especially for unsupervised speaker adaptation when the meta-learner is trained to perform unsupervised speaker adaptation. So far we were not able to outperform adaptation of LHUC parameters in TDNN models but we believe that the meta-learner ought to learn better adaptation strategy when properly trained.

In future work we want to focus on better understanding the training of the meta-learner. In this paper we used around 3 hours of data divided between 18 speakers to train the meta-learner. It is possible that because of this small training dataset the meta-learner is overfitting to speakers in the training data. Therefore, we plan to train the meta-learner with much larger training datasets to enable it to learn inter-speaker variability.

Another issue in training of the meta-learner is that the loss, as formulated in the Equation (4), does not penalise the meta-learner when it learns that the best strategy is to overfit to senones seen in the adaptation data. For example, when there is a big overlap in senones between two consecutive chunks the best strategy for the meta-learner is to predict only senones seen in the adaptation data. Therefore, we need to study how to regularise the meta-learner to not learn to overfit to adaptation data, either by using L1/L2 penalty on the difference of the adapted and the original weights [5] or by using Kullback-Leibler divergence between the posteriors of the adapted and the original model [6]. In preliminary experiments we tried to use L1 regularisation which forced the meta-learner to learn to adapt only some layers, but it did not improve the accuracy of the adapted model. Alternatively, we could use a lower entropy task for adaptation [15, 16], however we believe that a more expressive form of the meta-learner ought to figure out a relation between senones and monophones and co-adapt senones that correspond to the same phone.

Finally, it is necessary to explore ways of scaling the meta-learning both to larger models and to larger amounts of adaptation data. The problem with scaling to larger models is that it is not possible to fit a meta-learner to the GPU memory as the memory consumption is linear in the number of weights of an AM. Therefore, we would like to reparameterise the meta-learner to learn to perform some form of low-rank adaptation with lower memory requirements. Lower memory requirements would allow us to perform more adaptation steps without using truncated backpropagation through time.

5. Acknowledgements

This work was partially supported by the H2020 project SUMMA, under grant agreement 688139, and a PhD studentship funded by Bloomberg. We would like to thank Steve Renals for many insightful discussions.

6. References

[1] 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.

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

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

[4] 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.

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

[6] 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.

[7] N. Dehak, P. J. Kenny, R. Dehak, P. Dumouchel, and P. Ouellet, "Front-end factor analysis for speaker verification," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 19, no. 4, pp. 788–798, 2011.

[8] 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.

[9] Y. Huang and Y. Gong, "Regularized sequence-level deep neural network model adaptation," in *Interspeech*, 2015.

[10] K. Yao, D. Yu, F. Seide, H. Su, L. Deng, and Y. Gong, "Adaptation of context-dependent deep neural networks for automatic speech recognition," in *Spoken Language Technology Workshop (SLT), 2012 IEEE*. IEEE, 2012, pp. 366–369.

[11] Y. Zhao, J. Li, J. Xue, and Y. Gong, "Investigating online low-footprint speaker adaptation using generalized linear regression and click-through data," in *Acoustics, Speech and Signal Processing (ICASSP), 2015 IEEE International Conference on*. IEEE, 2015, pp. 4310–4314.

[12] 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.

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

[14] Z. Huang, S. M. Siniscalchi, I.-F. Chen, J. Wu, and C.-H. Lee, "Maximum a posteriori adaptation of network parameters in deep models," *arXiv preprint arXiv:1503.02108*, 2015.

[15] P. Swietojanski, P. Bell, and S. Renals, "Structured output layer with auxiliary targets for context-dependent acoustic modelling," in *Interspeech*, 2015.

[16] Z. Huang, J. Li, S. M. Siniscalchi, I.-F. Chen, J. Wu, and C.-H. Lee, "Rapid adaptation for deep neural networks through multi-task learning," in *Interspeech*, 2015.

[17] 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.

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

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

[20] A. Rousseau, P. Deléglise, and Y. Estève, "Enhancing the TED-LIUM corpus with selected data for language modeling and more TED talks," in *LREC*, 2014.

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

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

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

[24] 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.

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

[26] 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*, vol. 16, 2016, pp. 265–283.

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

[28] P. Bell, P. Swietojanski, J. Driesen, M. Sinclair, F. McInnes, and S. Renals, "The UEDIN ASR systems for the IWSLT 2014 evaluation," in *Proc. IWSLT*, 2014.

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