Differentiable Pooling for Unsupervised Acoustic Model Adaptation

Citation for published version:

Digital Object Identifier (DOI):
10.1109/TASLP.2016.2584700

Link:
Link to publication record in Edinburgh Research Explorer

Document Version:
Peer reviewed version

Published In:
IEEE/ACM Transactions on Audio, Speech, and Language Processing

General rights
Copyright for the publications made accessible via the Edinburgh Research Explorer is retained by the author(s) and / or other copyright owners and it is a condition of accessing these publications that users recognise and abide by the legal requirements associated with these rights.

Take down policy
The University of Edinburgh has made every reasonable effort to ensure that Edinburgh Research Explorer content complies with UK legislation. If you believe that the public display of this file breaches copyright please contact openaccess@ed.ac.uk providing details, and we will remove access to the work immediately and investigate your claim.
Differentiable Pooling for Unsupervised Acoustic Model Adaptation

Pawel Swietojanski, Member, IEEE, and Steve Renals, Fellow, IEEE

Abstract—We present a deep neural network (DNN) acoustic model that includes parametrised and differentiable pooling operators. Unsupervised acoustic model adaptation is cast as the problem of updating the decision boundaries implemented by each pooling operator. In particular, we experiment with two types of pooling parametrisations: learned $L_p$-norm pooling and weighted Gaussian pooling, in which the weights of both operators are treated as speaker-dependent. We perform investigations using three different large vocabulary speech recognition corpora: AMI meetings, TED talks and Switchboard conversational telephone speech. We demonstrate that differentiable pooling operators provide a robust and relatively low-dimensional way to adapt acoustic models, with relative word error rate reductions ranging from 5–20% with respect to unadapted systems, which themselves are better than the baseline fully-connected DNN-based acoustic models. We also investigate how the proposed techniques work under various adaptation conditions including the quality of adaptation data and complementarity to other feature- and model-space adaptation methods, as well as providing an analysis of the characteristics of each of the proposed approaches.

I. INTRODUCTION AND SUMMARY

Deep neural network (DNN) acoustic models have significantly extended the state-of-the-art in speech recognition [1] and are known to be able to learn significant invariances through many layers of non-linear transformations [2]. If the training and deployment conditions of the acoustic model are mismatched then the runtime data distribution can differ from the training distribution, bringing a degradation in accuracy, which may be addressed through explicit adaptation to the test conditions [2]–[9].

In this paper we explore the use of parametrised and differentiable pooling operators for acoustic adaptation. We introduce the approach of differentiable pooling using speaker-dependent pooling operators, specifically $L_p$-norm pooling and weighted Gaussian pooling (Section III), showing how the pooling parameters may be optimised by minimising the negative log probability of the class given the input data (Section IV), and providing a justification for the use of pooling operators in adaptation (Section V). To evaluate this novel adaptation approach we performed experiments on three corpora – TED talks, Switchboard conversational telephone speech, and AMI meetings – presenting results on using differentiable pooling for speaker independent acoustic modelling, followed by unsupervised speaker adaptation experiments in which adaptation of the pooling operators is compared (and combined) with learning hidden unit contributions (LHUC) [10], [11] and constrained/feature-space maximum likelihood linear regression (fMLLR) [12].

II. DNN ACOUSTIC MODELLING AND ADAPTATION

DNN acoustic models typically estimate the posterior distribution over a set of context-dependent tied states $s$ of a hidden Markov model (HMM) [13] given an acoustic observation $o$, $P(s|o) = \text{DNN}(o; \theta)$ [1], [14], [15]. The DNN is implemented as a nested function comprising $L$ processing layers (non-linear transformations):

$$\text{DNN}(o; \theta) = f^L (f^{L-1} (\ldots f^1 (o; \theta^1) \ldots; \theta^{L-1}); \theta^L) \quad (1)$$

The model is thus parametrised by a set of weights $\theta = \{ \theta^l \}_{l=1}^L$ in which the $l$th layer consists of a weight matrix and bias vector, $\theta^l = \{ W^l, b^l \}$, followed by a non-linear transformation $\phi$, acting on arbitrary input $x$:

$$f^l(x; \theta^l) = \phi^l (W^l x + b^l) \quad (2)$$

To form a probability distribution, the output layer employs a softmax transformation [16] $\phi^L(x) = \exp(x)/\sum_j \exp(x_j)$, whereas the hidden layer activation functions are typically chosen to be either sigmoid $\phi^l(x) = 1/(1 + \exp(-x))$ or rectified linear $\phi^l(x) = \max(0,x)$ units (ReLU) [17].

Yu et al [2] experimentally demonstrated that the invariance of the internal representations with respect to variabilities in the input space increases with depth (the number of layers) and that the DNN can interpolate well around training samples but fails to extrapolate if the data mismatch increases. Therefore one often explicitly compensates for unseen variabilities in the acoustic space.

Feature-space normalisation increases the invariance to unseen data by transforming the data such that it better matches the training data. In this approach the DNN learns an additional transform of the input features conditioned on the speaker or the environment. The transform, which is typically affine, is parametrised by an additional set of adaptation parameters. The most effective form of feature-space normalisation is constrained (feature-space) maximum-likelihood linear regression (fMLLR), referred to as fMLLR [12], in which the linear transform parameters are estimated by maximising the likelihood of the adaptation data under a Gaussian Mixture Model (GMM) / HMM acoustic model. To use fMLLR with a DNN acoustic model it is necessary to estimate a single input transform per speaker (using a trained GMM), using...
the resultant transformed data to train a DNN in a speaker adaptive training (SAT) manner. At runtime another set of fMLLR parameters is estimated for each speaker and the data transformed accordingly. This technique has consistently and significantly reduced the word error rate (WER) across several different benchmarks for both hybrid [1], [14] and tandem [18], [19] approaches. There are many successful examples of fMLLR adaptation of DNN acoustic models [1], [6], [8], [20]–[25]. One can also estimate the linear transform as an input layer of the DNN, often referred to as a linear input network (LIN) [3], [4], [6], [26]. LIN-based approaches have been mostly used in test-only adaptation schemes, whereas fMLLR requires SAT, but usually results in lower WERs.

Auxiliary feature approaches augment the acoustic feature vectors with additional speaker-dependent information computed for each speaker at both training and runtime stages – this is a form of SAT in which the model learns the distribution over tied states conditioned on some additional speaker-specific information. There has been considerable recent work exploring the use of i-vectors [27] for this purpose. I-vectors, which can be regarded as basis vectors spanning a subspace of speaker variability, were first used for adaptation in a GMM framework by Karafiat et al. [28], and were later successfully employed for DNN adaptation [29]–[34]. Other examples of auxiliary features include the use of speaker-specific bottleneck features obtained from a speaker separation DNN [35], the use of out-of-domain tandem features [24], as well as speaker codes [36]–[38] in which a specific set of units for each speaker is optimised. Kundu et al. [39] present an approach using auxiliary input features derived from the bottleneck layer of a DNN which is combined with i-vectors.

Model-based approaches adapt the DNN parameters using data from the target speaker. Liao [40] investigated this approach in both supervised and unsupervised settings using a few minutes of adaptation data. On a large DNN, when all weights were adapted, up to 5% relative improvement was observed for unsupervised adaptation, using a speaker-independent decoding to obtain DNN targets. Yu et al. [9] have explored the use of regularisation for adapting the weights of a DNN, using the Kullback-Liebler (KL) divergence between the output distributions produced by speaker-independent and the speaker-adapted models. This approach was also recently used to adapt parameters of sequence-trained models [41]. LIN may also be regarded as a form of model-based adaptation, and related approaches include adaptation using a linear output network (LON) or linear hidden network (LHN) [4], [7], [42].

Directly adapting all the weights of a large DNN is computationally and data intensive, and results in large speaker-dependent parameter sets. Smaller subsets of the DNN weights may be modified, including biases and slopes of hidden units [7], [34], [43], [44]. Another recently developed approach relies on learning hidden unit contributions (LHUC) for test-only adaptation [10], [11] as well as in a SAT framework [45]. One can also adapt the top layer using Bayesian methods resulting in a maximum a posteriori (MAP) approach [46], or address the sparsity of context-dependent tied-states when few adaptation data-points are available by modelling both monophones and context-dependent tied states using multi-task adaptation [47], [48] or a hierarchical output layer [49].

III. DIFFERENTIABLE POOLING

Building on our initial work [50], we present an approach to adaptation by learning hidden layer pooling operators with parameters that can be learned and adapted in a similar way to the other model parameters. The idea of feature pooling originates from Hubel and Wiesel’s pioneering study on visual cortex in cats [51], and was first used in computer vision to combine spatially local features [52]. Pooling in DNNs involves the combination of a set of hidden unit outputs into a summary statistic. Fixed poolings are typically used, such as average pooling (used in the original formulation of convolutional neural networks – CNNs) [53], [54] and max pooling (used in the context of feature hierarchies [55] and later applied to CNNs [56], [57]).

Reducing the dimensionality of hidden layers by pooling some subsets of hidden unit activations has become well investigated beyond computer vision, and the max operator has been interpreted as a way to learn piecewise linear activation functions – referred to as Maxout [58]. Maxout has been widely investigated for both fully-connected [59]–[61] and convolutional [62], [63] DNN-based acoustic models. Max pooling, although differentiable, performs a one-from-K selection, and hence does not allow hidden unit outputs to be interpolated, or their combination to be learned within a pool.

There have been a number of approaches to pooling with differentiable operators – differentiable pooling – a notion introduced by Zeiler and Fergus [64] in the context of constructing unsupervised feature extract for support vector machines in computer vision tasks. There has been some interest in the use of $L_p$-norm pooling with CNN models [57], [65] in which the sufficient statistic is the $p$-norm of the group of (spatially-related) hidden unit activations. Fixed order $L_p$-norm pooling was recently applied within the context of a convolutional neural network acoustic model [66], where it did not reduce the WER over max-pooling, and as an activation function in a fully-connected DNNs [67], where it was found to improve over maxout and ReLU.

A. $L_p$-norm ($\ell^p \rightarrow \ell^p$-$L_p$) pooling

In this approach we pool a set of activations using an $L_p$-norm. A hidden unit pool is formed by a set $R_k$ of $K$ affine projections which form the input to the $k$th pooling unit, which we write as an ordered set (vector) $a^k = \{w^T_i x + b_i\}_{i \in R_k}$. The output of the $k$th pooling unit is produced as an $L_p$ norm:

$$f_{L_p}(a^k; p_k) = ||a^k||_{p_k} = \left( \frac{1}{K} \sum_{i \in R_k} |a^k_i|^{p_k} \right)^{\frac{1}{p_k}}, \quad (3)$$

where $p_k$ is the learnable norm order for the $k$th unit, that can be jointly optimised with the other parameters in the model. To ensure that (3) satisfies a triangle inequality ($p_k \geq 1$; a necessary property of the norm), during optimisation $p_k$ is re-parametrised as $p_k = \zeta(p_k) = \max(1, p_k)$, where $p_k$ is the actual learned parameter. For the case when $p_k = \infty$ we obtain the max-pooling operator [55]:

$$||a^k||_\infty = \max \left( \{ |a^k_i| \}_{i \in R_k} \right). \quad (4)$$
The pooling contributions \( u(z^k; \vartheta^k) \) are normalised to sum to one within each pooling region \( R_k \) (7) and each weight \( u_i(z^k; \vartheta^k) \) is coupled with the corresponding value of \( z_i^k \) by a Gaussian kernel (8) (one per pooling unit) parameterised by the mean and precision, \( \vartheta^k = \{ \mu_k, \beta_k \} \):

\[
u_i(z^k; \vartheta^k) = \frac{v(z_i^k; \vartheta^k)}{\sum_{i' \in R_k} v(z_i^{k'}; \vartheta^k)},
\]

\[
v(z_i^k; \vartheta^k) = \exp \left( -\frac{\beta_k}{2} (z_i^k - \mu_k)^2 \right).
\]

Similar to \( L_p \)-norm pooling, this formulation allows a generalised pooling to be learned – from average (\( \beta \rightarrow 0 \)) to max (\( \beta \rightarrow \infty \)) – separately for each pooling unit \( f_G(z^k; \vartheta^k) \) within a model. The \textit{Diff-Gauss} model is thus parametrised by \( \theta_G = \{ \{ W^l, b^l, \mu^l, \beta^l, \eta^l \}_{l=1}^3, W^l, b^l \} \).


IV. LEARNING DIFFERENTIABLE POOLERS

We optimise the acoustic model parameters by minimising the negative log probability of the target HMM tied state given the acoustic observations using gradient descent and error back-propagation [70]; the pooling parameters may be updated in a speaker-dependent manner, to adapt the acoustic model to unseen data. In this section we give the necessary partial derivatives for \textit{Diff-} \( L_p \) and \textit{Diff-Gauss} pooling.

A. Learning and adapting \textit{Diff-} \( L_p \) pooling

In \textit{Diff-} \( L_p \) pooling we learn \( p_k \) which we express in terms of \( \rho, p_k = \zeta(\rho_k) \). Error back-propagation requires the partial derivative of the pooling region \( f_{L_p}(a^k; \rho_k) \) with respect to \( \rho_k \), which is given as:

\[
\frac{\partial f_{L_p}(a^k; \rho_k)}{\partial \rho_k} = \left( \sum_{i \in R_k} \log |a_i^k| \cdot |a_i^k|^{p_k-1} \right) \frac{\partial \rho_k}{\partial \rho_k} f_{L_p}(a^k; \rho_k),
\]

where \( \partial \rho_k/\partial \rho_k = 1 \) when \( p_k > 1 \) and 0 otherwise. The back-propagation through the norm itself is implemented as:

\[
\frac{\partial f_{L_p}(a^k; p_k)}{\partial a^k} = \frac{\partial f_{L_p}(a^k; p_k)}{\partial \rho_k} \frac{\partial \rho_k}{\partial a^k} = \frac{\partial f_{L_p}(a; p_k)}{\partial \rho_k} \frac{\partial \rho_k}{\partial a^k}.
\]

where \( \circ \) represents the element-wise Hadamard product, and \( G^k \) is a vector of \( f_{L_p}(a^k; p_k) \) activations repeated \( K \) times, so the resulting operation can be fully vectorised:

\[
G^k = \left[ f_{L_p}(a^k; p_k)^1, \ldots, f_{L_p}(a^k; p_k)^K \right]^\top.
\]

Normalisation by \( K \) in (3) is optional (see also Section VII-A) and the partial derivatives in (9) and (10) hold for the unnormalised case also: the effect of this is taken into account in the forward activation \( f_{L_p}(a^k; \rho_k) \).

Since (9) and (10) are not continuous everywhere, they need to be stabilised when \( \sum_{i \in R_k} |a_i^k|^{p_k} = 0 \). When computing logarithm in the numerator of (9) it is also necessary to ensure that each \( a_i^k > 0 \). In practice, we threshold each element to have at least a value \( \epsilon = 10^{-8} \) if \( a_i^k < \epsilon \). Note, this numerical stabilisation of \( a^k \) only applies to \( L_p \) units, not \textit{Diff-Gauss}.

---

**Fig. 1. Illustration of \textit{Diff-} \( L_p \) (a) and \textit{Diff-Gauss} (b) pooling operators.** See Sections III-A and III-B for further details and explanations of the symbols. LHUC scaling follows [68] and is used only during adaptation.
B. Learning and adapting Diff-Gauss pooling regions

To learn the Diff-Gauss pooling parameters $\theta^k = \{\mu^k, \beta^k\}$, we require the partial derivatives $\partial f_G(z^k)/\partial \mu_k$ and $\partial f_G(z^k)/\partial \beta_k$ to update pooling parameters, as well as $\partial f_G(z^k)/\partial z^k$ in order to back-propagate error signals to lower layers.

One can compute the partial derivative of (6) with respect to the input activations $z^k$ as:

$$\frac{\partial f_G(z^k)}{\partial z^k} = [(z^k) \cdot (J_u(v(z^k))J_v(z^k)) + u(z^k)]^\top,$$

where $J_u(v(z^k))$ is the Jacobian representing the partial derivative $\partial u(z^k)/\partial v(z^k)$:

$$J_u(v(z^k)) = \frac{\partial u(z^k)}{\partial v(z^k)} = \begin{bmatrix} \frac{\partial u(z^k_1)}{\partial v(z^k_1)} & \cdots & \frac{\partial u(z^k_1)}{\partial v(z^k_k)} \\ \vdots & \ddots & \vdots \\ \frac{\partial u(z^k_k)}{\partial v(z^k_1)} & \cdots & \frac{\partial u(z^k_k)}{\partial v(z^k_k)} \end{bmatrix},$$

whose elements can be computed as:

$$\frac{\partial u(z^k_i)}{\partial v(z^k_i)} = \left( \sum_{m \in R_k} v(z_m) \right)^{-1} (1 - u(z^k_i)),$$

and

$$\frac{\partial u(z^k_i)}{\partial v(z^k_i)} = \left( \sum_{m \in R_k} v(z_m) \right)^{-1} (-u(z^k_i)).$$

Likewise, $J_v(z^k)$ represents the Jacobian of the kernel function $v(z^k)$ in (8) with respect to $z^k$:

$$J_v(z^k) = \frac{\partial v(z^k)}{\partial z^k} = \begin{bmatrix} \frac{\partial v(z^k_1)}{\partial z^k_1} & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & \frac{\partial v(z^k_k)}{\partial z^k_k} \end{bmatrix},$$

and the elements of $J_v(z^k)$ can be computed as:

$$\frac{\partial v(z^k_i)}{\partial z^k_i} = -\beta_k (z^k_i - \mu_k) v(z^k_i).$$

Similarly, one can obtain the gradients with respect to the pooling parameters $\theta^k$. In particular, for $\beta_k$, the gradient is:

$$\frac{\partial f_G(z^k)}{\partial \beta_k} = \sum_{i \in R_k} [(z^k_1) \cdot (J_u(v(z^k))J_v(\beta_k))]_i,$$

where $J_v(\beta_k) = \partial v(z^k_i)/\partial \beta_k$ and $\partial v(z^k_i)/\partial \beta_k$ is:

$$\frac{\partial v(z^k_i)}{\partial \beta_k} = \frac{1}{2} (z^k_i - \mu_k)^2 v(z_i).$$

The corresponding gradient for $\partial f_G(z^k)/\partial \mu_k$ is obtained below (20). Notice, that $\partial v(z^k_i)/\partial z^k_i$ (17) and $\partial v(z^k_i)/\partial \mu_k$ (21) are symmetric, hence $J_u(\mu_k) = -J_v(z^k)$, and to compute $\partial f_G(z^k)/\partial \mu_k$ one can reuse the $(z^k) \cdot (J_u(v(z^k))J_v(z^k))$ term in (12), as follows:

$$\frac{\partial f_G(z^k)}{\partial \mu_k} = \sum_{i \in R_k} [(z^k_1) \cdot (J_u(v(z^k))J_v(\mu_k))]_i,$$

and

$$\frac{\partial v(z^k_i)}{\partial \mu_k} = -\frac{\partial v(z^k_i)}{\partial z^k_i} = \beta_k (z^k_i - \mu_k) v(z^k_i).$$

Fig. 2. Illustration of the representational efficiency and adaptation principles of an $L_p$ unit. (a) Unit circles as obtained under different norm $p$-orders. (b) An example decision boundary for two-class toy data (red and blue dots). The Diff–$L_p$ model is build out of one $L_p$ unit (with $K = 2$ linear inputs) and is able to draw highly non-linear decision-regions. (c) The model from (b) with $p = 1.0$ and (d) $p = \infty$. Red contours of the bottom two plots illustrate (e) the effect of adaptation of the origin (biases) of the linear inputs $a^k$ and (f) the effect of LHUC scaling. Further description in Section V. (Best viewed in colour.)

V. REPRESENTATIONAL EFFICIENCY OF POOLING UNITS

The aim of model-based DNN adaptation is to alter the learned speaker independent representation in order to improve the classification accuracy for data from a possibly mismatched test distribution. Owing to the highly distributed representations that are characteristic of DNNs, it is rarely clear which parameters should be adapted in order generalise well to a new speaker or acoustic condition.

Pooling enables decision boundaries to be altered, through the selection of relevant hidden features, while keeping the parameters of the feature extractors (the hidden units) fixed: this is similar to LHUC adaptation [68]. The pooling operators allow for a geometrical interpretation of the decision boundaries and how they will be affected by a constrained adaptation – the units within the pool are jointly optimised given the pooling parametrisation, and share some underlying relationship within the pool.

This is visualised for $L_p$ units in Fig. 2. Fig. 2 (a) illustrates
the unit circles obtained by solving $\|a^k\|_p = d$ for different orders $p$, with $d = 1.0$ and a pool of $K = 2$ linear inputs $a^k$. Such an $L_p$ unit is capable of closed-region decision boundaries, illustrated in Fig. 2 (b). The distance threshold $d$ is implicitly learned from data (through the $a^k$ parameters given $p$), resulting in an efficient representation [67], [69] compared with representing such boundaries using sigmoid units or ReLUs, which would require more parameters. Figs. 2 (c) and (d) show how those boundaries are affected when $p = 1$ (average pooling) and $p = \infty$ (max pooling), while keeping $a^k$ fixed. As shown in Section VII we found that updating $p$ is an efficient and relatively low-dimensional way to adjust decision boundaries such that the model’s accuracy on the adaptation data distribution improves.

It is also possible to update the biases (Fig. 2 (e), red contours) and the LHUC amplitudes (Fig. 2 (f), red contours). We experimentally investigate how each approach impacts adaptation WER in Section VII-B. Although models implementing Diff-Gauss units are theoretically less efficient in terms of SI representations compared to $L_p$ units, and comparable to standard fully-connected models, the pooling mechanism still allows for more efficient (in terms of number of SD parameters) speaker adaptation.

VI. EXPERIMENTAL SETUPS

We have carried out experiments on three corpora: the TED talks corpus [71] following the IWSLT evaluation protocol (www.iwslt.org); the Switchboard corpus of conversational telephone speech [72] (ldc.upenn.edu) and the AMI meetings corpus [73], [74] (corpus.amiproject.org). Unless explicitly stated otherwise, our baseline models share similar structure across the tasks – DNNs with 6 hidden layers (2,048 units per layer) using a sigmoid non-linearity. The output softmax layer models the distribution of context-dependent clustered tied states [75]. The features are presented in 11 (±5) frame long context windows. All the adaptation experiments, if not stated otherwise, were performed unsupervised using adaptation targets obtained from first-pass speaker-independent decoding of the corresponding SI system.

TED: The training data consisted of 143 hours of speech (813 talks) and the systems follow our previously described recipe [8]. However, compared to our previous work [8], [11], [50], our systems here make use of more accurate language models developed for our IWSLT-2014 systems [76]: in particular, the final reported results use a 4-gram language model estimated from 751 million words. The baseline TED acoustic models were trained on unadapted PLP features with first and second order time derivatives. We present results on four IWSLT test sets: dev2010, tst2010, tst2011 and tst2013 containing 8, 11, 8, and 28 talks respectively.

AMI: We follow a Kaldi GMM recipe [77] and use the individual headset microphone (IHM) recordings. On this corpus, we train the acoustic models using 40 mel-filter-bank (FBANK) features. We decode with a pruned 3-gram language model estimated from 800k words of AMI training transcripts interpolated with an LM trained on Fisher conversational telephone speech transcripts (1M words) [78].

Switchboard (SWBD): We follow a Kaldi GMM recipe [79], [80], using Switchboard–1 Release 2 (LDC97S62). Our baseline unadapted acoustic models were trained on MFCC features, while the SAT trained fMLLR variants utilise the usual Kaldi feature preprocessing pipeline, which is MFCC+LDA/MLLT+fMLLR. The results are reported on the full Hub5’00 set (LDC2002S09) – eval2000. eval2000 contains two types of data: Switchboard – which is better matched to the training data; and CallHome (CHE) English. Our reported results use 3-gram LMs estimated from the Switchboard and Fisher Corpus transcripts.

VII. RESULTS

A. Baseline speaker independent models

The structures of the differentiable pooling models were selected such that the number of parameters was comparable to the corresponding baseline DNN models, described in detail in [68]. For the Diff-–Lp and Diff-L2 types, the resulting models utilised non-overlapping pooling regions of size $K = 5$, with 900 $L_p$-norm units per layer. The Diff-Gauss models had pool sizes set to $K = 3$ (this was found to work best in our previous work [50]) which (assuming non-overlapping regions) results in 1175 pooling units per layer.

Training speaker independent Diff-L2 and Diff-Lp models: For both Diff-Lp and Diff-L2 we trained with an initial learning rate of .008 (for MFCC, PLP, FBANK features) and .006 (for fMLLR features). The learning rate was adjusted using the newbob learning scheme [81] based on the validation frame error rate. We found that applying explicit pool normalisation (dividing by $K$ in (3)) gives consistently higher error rates (typically an absolute increase of 0.3% WER): hence we used un-normalised $L_p$ units in all experiments. We did not apply post-layer normalisation [67]. Instead, we use max-norm approach – after each update we scaled the columns (i.e. each $a_i^k$) of the fully connected weight matrices such that their $L_2$ norms were below a given threshold (set to 1.0 in this work) [82]. For Diff-Lp models we initialised $p = 2.0$. Those parameters were optimised on TED and directly applied without further tuning for the other two corpora. In this work we have focussed on adaptation; Zhang et al [67] have reported further speaker independent experiments for fixed order $L_p$ units.

Training speaker independent Diff-Gauss models: The initial learning rate was set to 0.08 (regardless of the feature type), again adjusted using newbob. Initial pooling parameters were sampled randomly from normal distribution: $\mu \sim N(0,1)$ and $\beta \sim N(1,0.5)$. Otherwise, the hyperparameters were the same as for the baseline DNN models.

Baseline speaker independent results: Table I gives speaker independent results for each of the considered model.

1To be compatible with our previous adaptation work on Switchboard [45], [68] we are using the older set of Kaldi recipe scripts called $\pm 15$, and our baseline results are comparable with the corresponding baseline numbers previously reported. A newer set of improved scripts exists under $\pm 5$ which, in comparison to $\pm 15$, offer about 1.5% absolute lower WER.

2MFCC-Mel-frequency Cepstral Coefficients, LDA - Linear Discriminant Analysis, MLLT - Maximum Likelihood Linear Transform
types. The Diff-Gauss and Diff-$L_2$/Diff-$L_p$ models have comparable WERs, with a small preference towards Diff-$L_p$ in terms of the final WER on TED and AMI; all have lower average WER than the baseline DNN. The gap between the pooled models increases on AMI data where Diff-$L_p$ has a substantially lower WER (3.2% relative) than the other two models (Diff-Gauss and baseline DNN) by 2.1% relative.

Fig. 3 gives more insight into the Diff-$L_p$ models by showing how the final distributions of the learned order $p$ differ across AMI, TED and SWBD corpora. $p$ deviates more from its initialisation in the lower layers of the model; there is also a difference across corpora. This follows the intuition of how a multi-layer network builds its representation: lower layers are more dependent on acoustic variabilities, normalising for such effects, and hence feature extractors may differ across datasets – in contrast to the upper layers which rely on features abstracted away from the acoustic data. For these corpora, the order $p$ rarely exceeded 3, sometimes dropping below 2 – especially for layer 1 with SWBD data. However, most $L_p$ units, especially in higher layers, tend to have $p \sim 2$. This corresponds to previous work [67] in which fixed $L_p=2$ units tended to obtain lower WER. A similar analysis of Diff-Gauss pooling does not show large data-dependent differences in the learned pooling parameters.

**Training speed:** Table II shows the average training speeds for each of the considered models. Training pooling units is significantly more expensive than training baseline DNN models. This is to be expected as the pooling operations cannot be easily and fully vectorised. In our implementation training the Diff-Gauss or Diff-$L_p$ models is about 40% slower than training a baseline DNN. Not optimising $p$ during training (9) decreases the gap to about 20% slower. This indicates that training using fixed $L_2$ units, and then adapting the order $p$ in a speaker adaptive manner could make a good compromise.

### B. Adaptation experiments

We initially used the TED talks corpus to investigate how WERs are affected by adapting different layers in the model. The results indicated that adapting only the bottom layer brings the largest drop in WER; however, adapting more layers further improves the accuracy for both Diff-$L_p$ and Diff-Gauss models (Fig. 4 (a)). Since obtaining the gradients for the pooling parameters at each layer is inexpensive compared to the overall back-propagation, and adapting bottom layer gives largest gains, in the remainder of this work we adapt all pooling units. Similar trends hold when pooling adaptation is combined with LHUC adaptation, which on tst2010 improves the accuracies by 0.2–0.3% absolute.

Fig. 4 (b) shows WER vs. the number of adaptation iterations. The results indicate that one adaptation iteration is

---

**Table I**

<table>
<thead>
<tr>
<th>Model</th>
<th>TED tst2010</th>
<th>AMI eval</th>
<th>SWBD eval2000</th>
</tr>
</thead>
<tbody>
<tr>
<td>DNN</td>
<td>15.0</td>
<td>29.1</td>
<td>22.1</td>
</tr>
<tr>
<td>Diff-Gauss</td>
<td>14.6</td>
<td>29.0</td>
<td>21.4</td>
</tr>
<tr>
<td>Diff-$L_2$</td>
<td>14.6</td>
<td>28.5</td>
<td>21.3</td>
</tr>
<tr>
<td>Diff-$L_p$</td>
<td>14.5</td>
<td>27.6</td>
<td>21.3</td>
</tr>
</tbody>
</table>

**Table II**

<table>
<thead>
<tr>
<th>Model</th>
<th>#SD Parameters</th>
<th>TED</th>
<th>AMI</th>
<th>SWBD</th>
</tr>
</thead>
<tbody>
<tr>
<td>Diff-$L_p$</td>
<td>$L_p$ norm order</td>
<td>14.5</td>
<td>27.6</td>
<td>21.3</td>
</tr>
<tr>
<td>Diff-Gauss</td>
<td>$L_p$ norm order</td>
<td>14.6</td>
<td>29.0</td>
<td>21.4</td>
</tr>
<tr>
<td>Diff-$L_2$</td>
<td>$L_p$ norm order</td>
<td>14.6</td>
<td>28.5</td>
<td>21.3</td>
</tr>
<tr>
<td>Diff-$L_p$</td>
<td>$L_p$ norm order</td>
<td>14.5</td>
<td>27.6</td>
<td>21.3</td>
</tr>
</tbody>
</table>

**Table III**

<table>
<thead>
<tr>
<th>Model</th>
<th>#SD Parameters</th>
<th>TED</th>
<th>AMI</th>
<th>SWBD</th>
</tr>
</thead>
<tbody>
<tr>
<td>Diff-$L_p$</td>
<td>$L_p$ norm order</td>
<td>14.5</td>
<td>27.6</td>
<td>21.3</td>
</tr>
<tr>
<td>Diff-Gauss</td>
<td>$L_p$ norm order</td>
<td>14.6</td>
<td>29.0</td>
<td>21.4</td>
</tr>
<tr>
<td>Diff-$L_2$</td>
<td>$L_p$ norm order</td>
<td>14.6</td>
<td>28.5</td>
<td>21.3</td>
</tr>
<tr>
<td>Diff-$L_p$</td>
<td>$L_p$ norm order</td>
<td>14.5</td>
<td>27.6</td>
<td>21.3</td>
</tr>
</tbody>
</table>

**Fig. 3.** $L_p$ orders for the three corpora used in this work. Particular models share the same structure of hidden layers (the same number of $L_p$ units per layer – 900), though both dimensionality of the output layers as well as the acoustic features used to train each model, are different. Vertical black line at 2 denotes an initial $p$ setting of $L_p$ units.
TABLE IV

<table>
<thead>
<tr>
<th>Model</th>
<th>#SD Parameters</th>
<th>TED</th>
<th>AMI</th>
<th>SWBD</th>
</tr>
</thead>
<tbody>
<tr>
<td>Diff-Gauss</td>
<td>P(L−1)</td>
<td>12.8</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>+ LHUC</td>
<td>P(L−1)</td>
<td>13.1</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>+ Update μ</td>
<td>P(L−1)</td>
<td>13.1</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>+ Update β</td>
<td>P(L−1)</td>
<td>12.7</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>+ Update η</td>
<td>2P(L−1)</td>
<td>12.8</td>
<td>27.3</td>
<td>20.7</td>
</tr>
<tr>
<td>++ LHUC</td>
<td>3P(L−1)</td>
<td>12.5</td>
<td>27.0</td>
<td>20.4</td>
</tr>
<tr>
<td>++ Update μ, β</td>
<td>3P(L−1)</td>
<td>12.3</td>
<td>26.9</td>
<td>20.3</td>
</tr>
</tbody>
</table>

$L$ - #layers, $P$ - #pooling units in layer

sufficient and, more importantly, the model does not overfit when more iterations are used. This suggests that it is not necessary to regularise the model carefully (by Kullback-Leibler divergence [9], for instance) which is usually required when weights that directly transform the data are adapted. In the remainder, we adapt all models with a learning rate of 0.8 for three iterations (optimised on dev2010).

Table III shows the effect of adapting different pooling parameters (including LHUC amplitudes) for $L_p$ units. Updating only $p$, rather than any other stand-alone pooling parameter, gives a lower WER than LHUC adaptation with the same number of parameters (cf Fig. 2); however, updating both brings further reductions in WER. Adapting the bias is more data-dependent with a substantial increase in WER for SWBD; this also significantly increases the number of adapted parameters. Hence we adapted either $p$ alone, or $p$ with LHUC in the remaining experiments.

Table IV shows similar analysis but for Diff-Gauss model. For Diff-Gauss, it is beneficial to update both $\mu$ and $\beta$ (as in [50]), and LHUC was also found to be complementary. Notice, adapting with LHUC scalers is similar to altering $\eta$ in eq. (5) (assuming $\eta$ is tied per pool, as mentioned in Section III-B). As such, new parameters need not be introduced to adapt Diff-Gauss with LHUC as it is the case for Diff-$L_p$ units. In fact, last two rows of Table IV show that jointly updating $\mu$, $\beta$ and $\eta$ gives lower WER than updating $\mu$, $\beta$ and applying LHUC after pooling (see Fig. 1).

Analysis of Diff-$L_p$: Fig. 5 shows how the distribution of $p$ changes after the Diff-$L_p$ model adapts to each of the 28 speakers of tst2013. We plot the speaker independent histograms as well as the contours of the mean bin frequencies for each layer. For the adapted models the distributions of $p$ become less dispersed, especially in higher layers, which can be interpreted as shrinking the decision regions of particular $L_p$ units (cf Fig. 2). This follows the intuition that speaker adaptation involves reducing the variability that needs to be modelled, in contrast to the speaker independent model.

Taking into account the increased training time of Diff-$L_p$ models, one can also consider training fixed order Diff-$L_2$ [67], adapting $p$ using (9). The results in Fig. 5, as well as later results, cover this scenario. The adapted Diff-$L_2$ models display a similar trend in the distribution of $p$ to the Diff-$L_p$ models.

Analysis of Diff-Gauss: We performed a similar investigation on the learned Diff-Gauss pooling parameters (Fig. 6). In the bottom layers they are characterised by a large negative means and positive precisions which has the effect of turning off many units. After adaptation, some of them become more active, which can be seen based on shifted distributions of adapted pooling parameters in Fig. 6. The adaptation with Diff-Gauss has a similar effect as the

![Fig. 4. WER(%) on tst2010 as a function of a) number of bottom layers adapted with pooling operators and (optional) LHUC transforms and b) number of adaptation iterations](image-url)

![Fig. 5. Layer-wise histograms of learned $L_p$ orders (speaker independent) on TED. The vertical line (dashed-black) at 2 is the initial value of $p$; the black solid line denotes the mean contour ($\pm$ standard deviations in yellow) of the distribution of $p$ obtained after adaptation to 28 speakers of tst2013. Likewise, the dashed black line is the mean of the adapted $L_p$ orders ($\pm$ standard deviations in red) starting from a fixed-order Diff-$L_2$ speaker independent model. (Best viewed in colour.)](image-url)
adaptation of slopes and amplitudes [44], [83], but adapts \( K \) times fewer parameters.

**Amount of adaptation data and quality of targets:** We investigated the effect of the amount of adaptation data by randomly selecting adaptation utterances from tst2010 to give totals of 10s, 30s, 60s, 120s, 300s and more speaker-specific adaptation data per talker (Fig. 7 (a)). The WERs are an average over three independent runs, each sampling a different set of adaptation utterances (we did more passes in our previous work [11], [50], however, both LHUC and differentiable pooling operators were not sensitive to this aspect, resulting in small error bars between different results obtained with different random utterances). The Diff-\( L_p \) models offer lower WER and more rapid adaptation, with 10s of adaptation data resulting in a decrease in WER by 0.6% absolute (3.6% relative) which further increases up to 2.1% absolute (14.4% relative) when using all the speaker’s data in an unsupervised manner. Diff-Gauss is comparable in terms of WER to a DNN adapted with LHUC. In addition, both methods are complementary to LHUC adaptation, and to feature-space adaptation with fMMLLR (Tables VI and VII).

In order to demonstrate the modelling capacities of the different model-based adaptation techniques, we carried out a supervised adaptation (oracle) experiment in which the adaptation targets were obtained by aligning the audio data with reference transcripts (Fig. 7 (b)). We do not refine what the model knows about speech, nor the way it classifies it (the feature receptors and output layer are fixed during adaptation and remain speaker independent), but show that the re-composition and interpolation of these basis functions to approximate the unseen distribution of adaptation data is able to decrease the WER by 26.7% relative for Diff-\( L_p \) + LHUC scenario.

The methods are also not very sensitive to the quality of adaptation targets, they show very similar trends as LHUC, for which exact results for different qualities of adaptation targets resulting from re-scoring adaptation hypotheses with different language models were reported in [68].

**Summary:** Results for the proposed techniques are summarised in Tables V, VI, and VII for AMI, TED, and SWBD, respectively. The overall observed trends are as follows: (I) speaker independent pooling models return lower WERs than the baseline DNNs: Diff-Gauss < Diff-\( L_2 \) ≤ Diff-\( L_p \) (although the last two seem to be data-dependent); (II) the pooling models (Diff-Gauss, Diff-\( L_2 \) and Diff-\( L_p \)) are complementary to both fMMLLR and LHUC adaptation – as expected, the final gain depends on the degree of data mismatch; (III) one can effectively train speaker independent Diff-\( L_2 \) models and later alter \( p \) in a speaker dependent manner; (IV) the average relative improvement across all tasks with respect to baseline unadapted DNN models were 6.8% for Diff-Gauss, 9.1% for Diff-\( L_2 \) and 10.4% for Diff-\( L_p \); and (V) when comparing LHUC adapted DNN to LHUC adapted differentiable pooling models, the relative reductions in WER for the pooling models were 2%, 3.4% and 4.8% for Diff-Gauss, Diff-\( L_2 \) and Diff-\( L_p \), respectively.

**VIII. DISCUSSION AND CONCLUSIONS**

We have proposed the use of differentiable pooling operators with DNN acoustic models to perform unsupervised speaker adaptation. Differentiable pooling operators offer a relatively-low dimensional set of parameters which may be adapted in a speaker-dependent fashion.

We investigated the complementarity of differentiable pooling adaptation with two other approaches – model-based LHUC.
adaptation and feature-space fMLLR adaptation. We have not performed an explicit comparison with an i-vector approach to adaptation. However, some recent papers have compared i-vector adaptation with either LHUC and/or fMLLR on similar data which enables us some make indirect comparisons. For example, Samarakoon and Sim [34] showed that speaker-adaptive training with i-vectors gives a comparable results to test-only LHUC using TED data, and Miao et. al [33] suggested that LHUC is better than a standard use of i-vectors (as in Saon et al. [29]) on TED data, with a more sophisticated i-vector post-processing needed to equal LHUC. Since the proposed Diff-\(L_p\) and Diff-Gauss techniques resulted in WERs that were at least as good as LHUC (and were found to be complementary to fMLLR) we conclude that the proposed pooling-based adaptation techniques are competitive.

In the future, one could investigate extending the proposed techniques to speaker adaptive training (SAT) [84], [85], for example in a similar spirit as proposed in the context of SAT-LHUC [45]. In addition it would be interesting to investigate the suitability of adapting pooling regions in the framework of sequence discriminative training [79], [86], [87]. Our experience of LHUC in this framework [68], together with the observation that the pooling models are not prone to over-fitting in the case of small amounts of adaptation data, suggests that adaptation based on differentiable pooling is a promising technique for sequence trained models.

ACKNOWLEDGEMENT

The NST research data collection may be accessed at http://datashare.is.ed.ac.uk/handle/10283/786. This research utilised a K40 GPGPU board donated by NVIDIA Corporation. The authors would like to thank the reviewers for insightful comments that helped to improve the manuscript.

REFERENCES

of four research groups,” *IEEE Signal Processing Magazine*, vol. 29, no. 6, pp. 82–97, Nov 2012.


