Mixed Precision Low-Bit Quantization of Neural Network Language Models for Speech Recognition

Junhao Xu , Jianwei Yu , Shoukang Hu , *Member, IEEE*, Xunying Liu , *Member, IEEE*, and Helen Meng, *Fellow, IEEE*

Abstract—State-of-the-art language models (LMs) represented by long-short term memory recurrent neural networks (LSTM-RNNs) and Transformers are becoming increasingly complex and expensive for practical applications. Low-bit neural network quantization provides a powerful solution to dramatically reduce their model size. Current quantization methods are based on uniform precision and fail to account for the varying performance sensitivity at different parts of LMs to quantization errors. To this end, novel mixed precision neural network LM quantization methods are proposed in this paper. The optimal local precision choices for LSTM-RNN and Transformer based neural LMs are automatically learned using three techniques. The first two approaches are based on quantization sensitivity metrics in the form of either the KL-divergence measured between full precision and quantized LMs, or Hessian trace weighted quantization perturbation that can be approximated efficiently using matrix free techniques. The third approach is based on mixed precision neural architecture search. In order to overcome the difficulty in using gradient descent methods to directly estimate discrete quantized weights, alternating direction methods of multipliers (ADMM) are used to efficiently train quantized LMs. Experiments were conducted on state-of-the-art LF-MMI CNN-TDNN systems featuring speed perturbation, i-Vector and learning hidden unit contribution (LHUC) based speaker adaptation on two tasks: Switchboard telephone speech and AMI meeting transcription. The proposed mixed precision quantization techniques achieved "lossless" quantization on both tasks, by producing model size compression ratios of up to approximately 16 times over the full precision LSTM and Transformer baseline LMs, while incurring no statistically significant word error rate increase.

Index Terms—Language models, speech recognition, LSTM-RNN, transformer, low-bit quantization, ADMM.

I. INTRODUCTION

LANGUAGE models (LMs) are important components in automatic speech recognition (ASR) systems and many other applications. A key part of the statistical language modelling problem is to derive the suitable representation of long-range history contexts. Directly modelling long-span word histories using conventional back-off n-gram models [1] generally

leads to a severe data sparsity issue [2]. To this end, over the past few decades there have been significant efforts of developing artificial neural network based language modelling techniques in the speech technology community [3]–[14]. Neural network language models (NNLMs) representing longer span history contexts in a continuous and lower dimensional vector space, are used to improve the generalization performance. With the rapid progress of deep neural network (DNN) based ASR technologies in recent decades, the underlying network architectures of NNLMs have evolved from feedforward structures [3]–[7] to more advanced variants represented by long-short term memory recurrent neural networks (LSTM-RNNs) [8]–[10], [15] and more recently neural Transformers [11]–[14], [16] that are designed for modelling longer range contexts. In particular, Transformer based language models in recent years have defined state-of-the-art performance across a range of ASR task domains [11]–[14], [17]. These models [11]–[13], [17] are often constructed using a deep stacking of multiple self-attention based neural building blocks [18]–[20], each of which also includes residual connections [21] and layer normalization modules [22]. Additional positional encoding layers [16], [23] are employed to augment the self-attention structures with word sequence order information. Performance improvements over conventional LSTM-RNN language models have been widely reported [11], [24].

However, the increasingly deeper and more complex architecture designs featuring in LSTM-RNN and Transformer models present many challenges for current ASR technologies. These not only lead to a large increase in their overall memory footprint and computational cost when operating on the cloud, but also creates difficulty when deployed on edge devices to enhance privacy and reduce latency. In a wider context within the speech technology community, such dramatically increasing demand for computational resources is consistent with the recent trend of moving towards a data and computational intensive all neural end-to-end (E2E) modelling paradigm represented by, for example, Transformers [25]–[27], RNN transducers (RNN-T) [28], and listen, attend and spell (LAS) [29]. State-of-the-art ASR systems featuring these end-to-end approaches often contain a very large number of parameters, for example, up to 280 million [30]. Hence, there is a pressing need of developing ultra-compact, low footprint language modelling methods, and ASR technologies in general, to facilitate more aggressive reduction in memory footprint, model training and evaluation time while maintaining competitive accuracy performance.

To this end, significant efforts have been made in both the machine learning and speech technology communities to develop DNN compression techniques [31]–[74]. Pruning based methods exploiting the structural and parameter sparsity were used to reduce DNN model size in [38]–[44]. They are particularly useful for models containing large fully connected layers such as the ResNet systems [43], [44]. Knowledge distillation and teacher-student learning [45]–[47] based approaches extract information from a pre-trained, larger model into a smaller one. Low rank matrix factorization [48]–[51], [55], and neural architecture search (NAS) based methods [63]–[72] have also been proposed.

Another powerful family of techniques recently drawing increasing interest across the machine learning, computer vision and speech technology communities to solve this problem is to use low-bit DNN quantization techniques [31]–[37], [52], [57], [58], [62], [74], [75]. By replacing floating point based DNN parameters with low precision values, for example, binary numbers, model sizes can be dramatically reduced without changing the DNN architecture [32], [57], [73]. Further DNN size reduction can be obtained when low-precision quantization is used in combination with neural architecture search (NAS) techniques, for example, in the SqueezeNet system designed for computer vision tasks [52]. In contrast to the extensive prior research works on low-bit quantization methods primarily targeting computer vision tasks [31]–[37], [52], only limited previous research in this direction has been conducted in the context of language modelling [57], [58] and ASR systems [56], [59].

Two issues are associated with current low-bit DNN quantization methods. First, these approaches are predominantly based on uniform precision, where an identical bit-width is applied to all weight parameters for quantization [76]. Within such framework, the varying local performance sensitivity exhibited at different parts of the underlying DNN system to quantization errors is not taken into account. In practice, this often leads to large performance degradation against full precision models. Second, when DNN weights are restricted to discrete values, the conventional back-propagation (BP) algorithm based on gradient descent methods cannot be directly applied to estimate the quantized model parameters. Existing approaches for training quantized DNNs often use a modified BP algorithm [31], [32]. In this approach, low precision quantized parameters are first used in a forward pass to compute the error loss, before full precision parameters are then used in a backward pass to propagate the gradients for subsequent model update. However, the inconsistency between quantized, discrete weights and the SGD algorithm assuming continuous and differentiable error cost functions leads to not only very slow convergence in training, but also performance degradation against full precision models.

In order to address the first issue discussed above regarding performance sensitivity, and motivated by the recent development of mixed precision DNN acceleration hardware that allows multiple locally selected precision settings to be used [37], novel mixed precision DNN quantization approaches are proposed in this paper by utilizing locally variable bit-widths at different layer components of LSTM-RNN and Transformer LMs. The

optimal local precision settings are automatically learned using three techniques. The first two approaches are based on quantization sensitivity metrics in the form of either Hessian trace weighted quantization perturbation that can be approximated efficiently via matrix free techniques, or the KL-divergence measured between full precision and quantized language models. The third approach is based on mixed precision neural architecture search.

In order to address the second issue over the difficulty in using gradient descent methods to directly estimate NNLMs of discrete weights, the general problem of estimating quantized DNN model parameters is reformulated as an optimization task. For any form of quantized NNLMs using uniform or mixed precision settings, alternating direction methods of multipliers (ADMM) [17], [73], [77] are proposed to efficiently train their discrete parameters. Two sets of model parameters respectively associated with a full precision neural network LM, and the corresponding optimal quantized model with a particular precision setting, are iteratively learned via a decomposed dual ascent scheme in an alternating fashion. This novel quantized NNLM estimation algorithm draws strength from both the decomposability of classic dual ascent schemes and the stable convergence of multiplier methods.

Experiments were conducted on state-of-the-art LF-MMI CNN-TDNN and TDNN systems [78] featuring speed perturbation, i-Vector [79] and learning hidden unit contribution (LHUC) based speaker adaptation [80] on two tasks, Switchboard telephone speech [81] and AMI meeting transcription [82]. Experimental results suggest the proposed mixed precision LM quantization techniques achieved model size compression ratios of about 16 times over the full precision Transformer LM baselines with no statistically significant recognition performance degradation.

The main contributions of this paper are summarized as follows.

1) To the best of our knowledge, this paper presents the first work in the speech technology community to apply mixed precision DNN quantization techniques to both LSTM-RNN and Transformer based NNLMs. In contrast, prior researches within the speech community in this direction largely focused on uniform precision based quantization of convolutional neural networks (CNNs) acoustic models [62] and LSTM-RNN language models [57], [58], [75].

2) To the best of our knowledge, this paper is the first work to introduce ADMM based neural network quantization techniques for speech recognition tasks. In contrast, prior researches with the speech technology community on low-bit quantization of CNNs [62] and LSTM-RNN LMs [57], [58], [75] used the modified BP algorithm [31], [32] while the inconsistency between discrete, quantized parameters and gradient based SGD update remains unaddressed.

3) Transformer LM model size compression ratios of up to approximately 16 times over the full precision Transformer LM baselines with no statistically significant recognition performance degradation were obtained using an average 2-bit mixed precision configuration. To the best of our knowledge, this is the best low-bit Transformer language model compression ratio

published so far in the speech technology community while incurring no recognition accuracy loss.

The rest of the paper is organized as follows. LSTM-RNN and Transformer based NNLMs are reviewed in Section II. A general neural network quantization scheme based on uniform or locally varying mixed quantization precisions are presented in Section III. ADMM based training of quantized NNLMs are presented in Section IV. Section V presents three novel mixed precision quantization methods. Experiments and results are shown in Section VI. Finally, conclusions and possible future work are discussed in Section VII.

II. NN LANGUAGE MODELS

This section reviews two types of neural network language models that are widely used in state-of-the-art speech recognition systems: long-short term memory recurrent neural network (LSTM-RNN) and Transformer based language models.

A. Recurrent Neural Network LMs

The form of LSTM-RNN language models considered in this paper computes the word probability $\mathbf{w}_t$ given the preceding history context of $t-1$ words $\mathbf{w}_1, \ldots, \mathbf{w}_{t-1}$ as

$$P(\mathbf{w}_t|\mathbf{w}_1^{t-1}) \approx P(\mathbf{w}_t|\mathbf{w}_{t-1}, \mathbf{h}_{t-1}), \tag{1}$$

where $\mathbf{h}_{t-1}$ is the D-dimensional vector *hidden state* encoding part of the history information $(\mathbf{w}_1, \ldots, \mathbf{w}_{t-2})$ up to word $\mathbf{w}_{t-2}$, where D is the number of RNNLM hidden layer nodes. The most recent word history $\mathbf{w}_{t-1}$ is represented by a N-dimensional one-hot vector $\tilde{\mathbf{w}}_t$, where N is the vocabulary size. This one-hot word input vector is first projected into a M-dimensional $(M \ll N)$ linear vector embedding as

$$\mathbf{x}_t = \mathbf{\Theta}_U \tilde{\mathbf{w}}_t^\top, \tag{2}$$

where $\mathbf{\Theta}_U$ is a projection matrix to be learned, before being further fed into the hidden layers where non-linear transformations are applied. The hidden state $\mathbf{h}_t$ is calculated within the LSTM cells [83] where the previous hidden state $\mathbf{h}_{t-1}$ and the current word input embedding $\tilde{\mathbf{w}}_t$ is combined. Each LSTM memory cell consists of a set of Sigmoidal gating activations, the input gate $\mathbf{i}_t$, forget gate $\mathbf{f}_t$, cell gate $\tilde{\mathbf{c}}_t$ and output gate $\mathbf{o}_t$. These are used to control the information flow within the cells in order to trap longer range history contexts and address the vanishing gradient issue. The respective outputs from these gates are given by

$$\mathbf{f}_t = \sigma\left(\mathbf{\Theta}_{\mathrm{f}}\left[\mathbf{x}_{t-1}, \mathbf{h}_{t-1}, 1\right]^\top\right) \tag{3}$$

$$\mathbf{i}_t = \sigma\left(\mathbf{\Theta}_{\mathrm{i}}\left[\mathbf{x}_{t-1}, \mathbf{h}_{t-1}, 1\right]^\top\right) \tag{4}$$

$$\tilde{\mathbf{c}}_t = \tanh\left(\mathbf{\Theta}_{\mathrm{c}}\left[\mathbf{x}_{t-1}, \mathbf{h}_{t-1}, 1\right]^\top\right) \tag{5}$$

$$\mathbf{o}_t = \sigma\left(\mathbf{\Theta}_{\mathrm{o}}\left[\mathbf{x}_{t-1}, \mathbf{h}_{t-1}, 1\right]^\top\right) \tag{6}$$

where $\tanh(\mathbf{u}) = [\tanh(\mathbf{u}_1), \ldots, \tanh(\mathbf{u}_D)]$ for any $\mathbf{u} \in \mathbf{R}^D$.

The final hidden state is normally computed recursively in an auto-regressive fashion, for example, from left to right in case of standard uni-directional LSTM-RNN LMs modelling history contexts only [5], [8], [10]. Using the above four gating functions outputs, the LSTM-RNN cell contents $\mathbf{c}_t$ and hidden state representation $\mathbf{h}_t$ are finally computed as

$$\mathbf{c}_t = \mathbf{f}_t \otimes \mathbf{c}_{t-1} + \mathbf{i}_t \otimes \tilde{\mathbf{c}}_t \tag{7}$$

$$\mathbf{h}_t = \mathbf{o}_t \otimes \tanh(\mathbf{c}_t), \tag{8}$$

where $\otimes$ is the Hadamard product.

B. Transformer LMs

The Transformer model architecture considered in this paper features a deep stacking of multiple Transformer decoder blocks. Unless otherwise stated, 6 Transformer decoder blocks are used in all the experiments of this paper. As shown in the top part of Fig. 1, each Transformer decoder block consists of a multi-head self-attention [18]–[20] module and a feed forward module. Residual connections [21] and layer normalization operations [22] are also inserted between these two modules. Let $\mathbf{x}_t^{l-1}$ denotes the output of the $(l-1)$-th Transformer block at input word position t. The multi-head self-attention module in the succeeding l-th block transforms $\mathbf{x}_t^{l-1}$ to $\mathbf{z}_t^l$ as follows:

$$\mathbf{q}_t^l, \mathbf{k}_t^l, \mathbf{v}_t^l = \mathbf{\Theta}_Q^l \mathbf{x}_t^{l-1}, \mathbf{\Theta}_K^l \mathbf{x}_t^{l-1}, \mathbf{\Theta}_V^l \mathbf{x}_t^{l-1} \tag{9}$$

$$\mathbf{y}_t^l = \mathrm{Attention}(\mathbf{k}_1^l, \ldots, \mathbf{k}_t^l, \mathbf{v}_1^l, \ldots, \mathbf{v}_t^l, \mathbf{q}_t^l) \tag{10}$$

$$\mathbf{z}_t^l = \mathbf{\Theta}_h^l \mathbf{y}_t^l + \mathbf{x}_t^{l-1} \tag{11}$$

$$\mathbf{o}_t^l = \mathrm{LayerNorm}(\mathbf{z}_t^l) \tag{12}$$

where $\mathbf{\Theta}_Q^l, \mathbf{\Theta}_K^l, \mathbf{\Theta}_V^l$ denote the learnable query, key, value projection matrices which map $\mathbf{x}_t^{l-1}$ into the corresponding vector representations of query $\mathbf{q}_t^l$, key $\mathbf{k}_t^l$ and value $\mathbf{v}_t^l$ respectively. $\mathbf{z}_t^l$ is the sequence of of cached key-value vector pairs up to time t, which only contains the history context information and can prevent the model from using any future context. Attention$(\cdot)$ denotes the scaled multi-head dot product self-attention [20]. LayerNorm$(\cdot)$ represents the layer normalization operation [22]. $\mathbf{\Theta}_h^l$ denotes the learnable projection matrix applied to the outputs of the Attention operation prior to layer normalization. The normalized output $\mathbf{o}_t^l$ is then fed into the feed forward module:

$$\mathbf{s}_t^l = \mathbf{\Theta}_2^l \mathrm{GELU}(\mathbf{\Theta}_1^l \mathbf{o}_t^l) + \mathbf{o}_t^l \tag{13}$$

$$\mathbf{x}_t^l = \mathrm{LayerNorm}(\mathbf{s}_t^l) \tag{14}$$

where $\mathbf{\Theta}_1^l$ and $\mathbf{\Theta}_2^l$ are the weight matrices that are applied to the normalized output $\mathbf{o}_t^l$ before Gaussian error linear unit (GELU) [84] activation functions and after. For simplicity the bias vectors optionally used in the feed forward modules are omitted in the above Equation (13). In addition, positional embedding layers are also used in all the transformer LMs considered in this paper.

III. NEURAL NETWORK QUANTIZATION

The standard n-bit quantization problem for deep neural networks (DNNs) considers the task of for any full precision weight parameter, Θ, finding its closest discrete approximation from the

following quantization table with a global scaling factor α and quantization parameter V,

$$q = \alpha V \in \{0, \pm\alpha, \ldots, \pm\alpha \cdot (2^{n-1} - 1)\} \quad (15)$$

as the one that incurs the minimum quantization error

$$f(\Theta) = \arg\min_q |\Theta - q| \quad (16)$$

Further simplification to the above quantization table of Equation (15) leads to extremely low bit quantization based on, for example, binary values $\{-1, 1\}$ [34], [74], or tertiary values $\{-1, 0, 1\}$ [85].

It is assumed in the above standard quantization process that a global quantization table is applied to all weight parameters. In order to account for the fine-grained local quantization sensitivity, the following more general form of quantization is considered for a particular model parameter $\Theta^{(l)}$ within any of the l-th weight cluster, for example, all parameters of the same LSTM or Transformer LM layer,

$$f(\Theta^{(l)}) = \arg\min_{Q^{(l)}} |\Theta^{(l)} - Q^{(l)}| \quad (17)$$

can be used. The locally shared l-th quantization table is

$$Q^{(l)} = \alpha^{(l)} V^{(l)} \in \{0, \pm\alpha^{(l)}, \ldots, \pm\alpha^{(l)}(2^{n_l - 1} - 1)\} \quad (18)$$

where the full precision scaling factor $\alpha^{(l)}$ is used to adjust the dynamic range of all the quantized weights in the l-th cluster. It is shared locally within each individual DNN parameter clusters. The locally variable quantization bit length n_l can be set to be $1, 2, 4, 8$ etc. depending on the optimal precision settings to be used. A special case, when the local quantization table in Equation (18) is shared across all the layers in the system, this leads to the traditional uniform precision quantization. The above tying of quantization tables may be flexibly performed at either layer, or node level, or in the extreme case at individual parameter level (this is equivalent to no quantization being applied). Intuitively, the longer the local quantization precision bit widths $\{n_l\}$ are used at each part of the underlying neural language model, a smaller compression ratio after quantization and reduced performance degradation is expected.

IV. ADMM BASED TRAINING OF QUANTIZED DNN

One major challenge faced by both uniform and mixed precision quantization is that the gradient descent methods and back-propagation (BP) algorithm cannot be directly used when weights are quantized to discrete values. To this end, mixed precision BP was proposed in [31], [32] where low precision binarized parameters were first used in the forward pass to compute the error loss before full precision parameters are used in the backward pass to propagate the gradients. However, there is an inconsistency between quantized, discrete weights and the assumption over continuous and differentiable error cost functions in SGD update. Hence, directly training quantized system using mixed precision BP leads to very slow convergence and the performance gap between full precision and quantized systems remains large. An alternative solution to this problem is

to reformulate quantization as a constrained optimization problem solved by the alternating direction methods of multipliers (ADMM) [73].

The ADMM based optimization decomposes a dual ascent problem into alternating updates of two variables. In the context of the neural network LM quantization problem considered here, these correspond to the full precision model weights update and the discrete quantization tables estimation. The overall ADMM Lagrange function is given as

$$L = \mathcal{F}_{ce}(\Theta) + (\gamma\lambda)^\top \cdot (\Theta - f(\Theta)) + \frac{\gamma}{2}||\Theta - f(\Theta)||_2^2 \quad (19)$$

where $\mathcal{F}_{ce}$ is the cross entropy (CE) loss, Θ are the full precision model parameters. $f(\Theta)$ represents the quantization of the parameters calculated from the quantization using Equation (17). $\gamma > 0$ is the penalty parameter which is empirically set as 10^{-3} throughout this paper and λ denotes the Lagrangian multiplier. The standard Lagrangian term expressed in the form of a dot product between the multiplier variable λ and the quantization error, $\Theta - f(\Theta)$, is shown as the second term in Equation (19). In order to further improve the robustness and convergence speed of the ADMM algorithm, an additional term related to the quantization error squared norm, shown as the third term in Equation (19), is also introduced to form an augmented Lagrangian [73]. Further rearranging Equation (19) leads to the following loss function.

$$L = \mathcal{F}_{ce}(\Theta) + \frac{\gamma}{2}||\Theta - f(\Theta) + \lambda||_2^2 - \frac{\gamma}{2}||\lambda||^2 \quad (20)$$

Given a particular uniform or mixed precision quantization configuration, the ADMM algorithm is iteratively performed to find the optimal scaling factors, $\{\alpha^{(l)}\}$, in the quantization table(s) of Equation (18). For simplicity, in the following detailed description of the algorithm, we assume a globally shared quantization table $\{0, \pm\alpha, \ldots, \pm\alpha(2^{n-1} - 1)\}$ is applied to all parameters and a single scaling factor α is to be learned. The following iterative update can be extended when multiple shared quantization tables of different bit-widths n_l in Equation (18) are used. When the ADMM algorithm is performed at the $(k + 1)^{th}$ iteration, the overall update can be split into three stages presented in the following subsections.

A. Full Precision Model Parameter Update

The following equation is used to update the full precision weight parameters $\Theta^{(k+1)}$.

$$\Theta^{(k+1)} = \arg\min_\Theta L(\Theta, f(\Theta^{(k)}), \lambda^{(k)}) \quad (21)$$

where $f(\Theta^{(k)}), \lambda^{(k)}$ are the quantized weights and error variable at the kth iteration. The gradient of the loss function in Equation (20) w.r.t Θ is calculated as the following.

$$\nabla L = \nabla\mathcal{F}_{ce} + \gamma(\Theta - f(\Theta^{(k)}) + \lambda^{(k)}) \quad (22)$$

It is found in practice that the quadratic term of the augmented Lagrangian of Equation (20) can dominate the loss function computation and lead to a local optimum. Following the extra-gradient method suggested in [86], the solution to address this

---

Algorithm 1: Quantization Variables Update Inner Loop.

Result: $\alpha^{(k+1)}$, $\boldsymbol{V}_\alpha^{(k+1)}$

$j = 0$;
$\alpha_0^{(k+1)} = \alpha^{(k)}$;
$\boldsymbol{V}_{0,\alpha}^{(k+1)} = \arg\min_{\boldsymbol{V}} |\boldsymbol{\Theta}^{(k+1)} - \alpha_0^{(k+1)}\boldsymbol{V}|$;

while $j < 20$ **&&** *convergence is not reached* **do**

$\quad \alpha_{j+1}^{(k+1)} = \dfrac{(\boldsymbol{\Theta}^{(k+1)}+\boldsymbol{\lambda}^{(k)})^\top \boldsymbol{V}_{j,\alpha}^{(k+1)}}{\boldsymbol{V}_{j,\alpha}^{(k+1)\top}\boldsymbol{V}_{j,\alpha}^{(k+1)}}$;

$\quad \boldsymbol{V}_{j+1,\alpha}^{(k+1)} = \arg\min_{\boldsymbol{V}} |\boldsymbol{\Theta}^{(k+1)} - \alpha_{j+1}^{(k+1)}\boldsymbol{V}|$;

$\quad j = j + 1$;

end

$\alpha^{(k+1)} = \alpha_j^{(k+1)}$;
$\boldsymbol{V}_\alpha^{(k+1)} = \boldsymbol{V}_{j,\alpha}^{(k+1)}$

---

issue in this paper is to perform the gradient calculation by one additional step ahead to improve the convergence.

$$\bar{\boldsymbol{\Theta}} \leftarrow \boldsymbol{\Theta}^{(k)} - \eta_1 \nabla L(\boldsymbol{\Theta})$$

$$\boldsymbol{\Theta}^{(k+1)} \leftarrow \boldsymbol{\Theta}^{(k)} - \eta_2 \nabla L(\bar{\boldsymbol{\Theta}}) \tag{23}$$

Here $\bar{\boldsymbol{\Theta}}$ represents the temporary variable used to store the intermediate updated parameters, and η_1 and η_2 are separate learning rates that are empirically set as 0.02 and 0.001 throughout the experiments of this paper.

B. Quantization Variables Update

The quantization variables including the scaling factor α in a globally shared quantization table $\{0, \pm\alpha, \ldots, \pm\alpha(2^{n-1} - 1)\}$, and the corresponding quantized parameters derived using Equation (17) can be solved by minimizing the following:

$$\min_f ||\boldsymbol{\Theta}^{(k+1)} - f(\boldsymbol{\Theta}^{(k)}) + \boldsymbol{\lambda}^{(k)}||_2^2$$

$$\Rightarrow \min_{\alpha, \boldsymbol{V}} ||\boldsymbol{\Theta}^{(k+1)} + \boldsymbol{\lambda}^{(k)} - \alpha^{(k)}\boldsymbol{V}_\alpha^{(k)}||^2 \tag{24}$$

where alternating updates of the scaling factor estimate $\alpha^{(k)}$ and the associated quantized model parameters $\boldsymbol{V}_\alpha^{(k+1)}$ are performed via an inner-loop within the current $(k+1)^{th}$ outer iteration.

The following algorithm shows the details of such inner-loop update. In all the experiments of this paper, the maximum number of inner-loops is set as 20 and practically found sufficient to ensure convergence.

The minimization operation of Equation (24) aims to find the corresponding quantized model parameters given the full precision parameter updates in Section IV-A. As it is non-trivial to update both the scaling parameters $\alpha^{(l)}$ of the quantization table of Equation (18) and the quantized parameters derived using Equation (17) at the same time, an alternating estimation procedure is used here in the inner-loop of Algorithm 1, to produce interleaving updates of the scaling parameters $\alpha^{(l)}$ when fixing the current quantized parameters $\boldsymbol{V}_\alpha^{(k)}$, and vice versa when updating the quantized parameters $\boldsymbol{V}_\alpha^{(k)}$, while keeping

$\alpha^{(l)}$ unchanged. Intuitively the resulting update of $\alpha^{(l)}$ accounts for the change in the dynamic range of model parameters before and after quantization.

C. Quantization Error Update

The Lagrange multiplier variable $\boldsymbol{\lambda}$, now encoding the accumulated quantization errors computed at each iteration, is updated as

$$\boldsymbol{\lambda}^{(k+1)} = \boldsymbol{\lambda}^{(k)} + \boldsymbol{\Theta}^{(k+1)} - f(\boldsymbol{\Theta}^{(k+1)}) \tag{25}$$

In all experiments of this paper the scaling factors $\alpha^{(l)}$ in Equation (18) are initialized to 1.0 before the ADMM update is performed.

The above ADMM estimation of quantized neural network LMs can be executed iteratively until convergence measured in terms of validation data perplexity. In practice, a maximum number of 20 ADMM iterations was used throughout all the experiments of this paper to obtain convergence for all quantized LSTM-RNN and transformer LMs, as will be shown later in the convergence speed analysis of the following Section VI of experiments.

V. MIXED PRECISION QUANTIZATION

This section presents three approaches to automatically learn the optimal local precision settings previously introduced in Equation (18) for DNN quantization. The first two minimizes the performance sensitivity to low-bit quantization. They are measured using either the KL divergence between full precision and quantized LMs, or the log-likelihood curvature with respect to quantization error. The third approach uses a mixed precision differentiable neural architecture search technique. Examples of their application to Transformer and LSTM-RNN based LMs are also given.

A. KL Divergence Based Mixed Precision Quantization

The ultimate goal for any DNN quantization task, including the neural network LMs considered in this paper, is to obtain a "lossless" model compression such that the distance between the distribution embodied by the original full precision LM and that of the quantized model must be minimized. This requires the relative entropy, or equivalently the Kullback-Leibler (KL) divergence between full precision and quantized neural network LMs to be minimized. As a special case, the KL divergence is zero when a lossless compression is achieved, so that the same LM distribution is preserved after quantization. The use of KL divergence based distance metrics in early researches led to widely adopted back-off n-gram LM pruning techniques [87], [88].

In this paper, the KL divergence between the probability distributions obtained from a full precision NNLM and its quantized counterpart using a particular mixed precision setting is used to measure the resulting performance sensitivity. Taking a L-layer Transformer LM for example, for any quantization $f(\cdot)$ being applied to the full precision parameters $\boldsymbol{\Theta}$, the KL divergence based quantization sensitivity measure is computed over the

training data of N_w words as,

$$\Omega^{\text{KL}} = \sum_{i=1}^{L} \Omega_i^{\text{KL}} = \sum_{i=1}^{L} D_{\text{KL}}(P(\boldsymbol{\Theta_i}) || P(f_{n_i}(\boldsymbol{\Theta_i})))$$

$$= \sum_{i=1}^{L} \sum_{t=1}^{N_w} P(w_t|w_{t-1}, h_{t-1}, \boldsymbol{\Theta_i}) \ln \frac{P(w_t|w_{t-1}, h_{t-1}, \boldsymbol{\Theta_i})}{P(w_t|w_{t-1}, h_{t-1}, f_{n_i}(\boldsymbol{\Theta_i}))}$$
(26)

where $\boldsymbol{\Theta}_i$ denote the full precision parameters of the ith layer, and $f_{n_i}(\boldsymbol{\Theta}_i)$ its associated n_i-bit quantized parameters given a particular local precision bit width n_i.

Given a target average quantization precision such as 2-bit, the local quantization bit widths used in each layer should be selected such that the total performance sensitivity in Equation (26) is minimized while satisfying the target model size constraint. However, directly evaluate the combined KL divergence metric for all possible mixed precision local quantization settings leads to a very large number of possible systems to be considered. For example, choosing among 4 different precision settings, 1-bit, 2-bit, 4-bit and 8-bit, across all 6 layers of a Transformer LM, produces a total of $4^6 = 4096$ mixed precision quantized Transformer LMs to be evaluated in terms of KL divergence against the full precision model.

In order to address this scalability issue, a practical implementation adopted in this paper is based on a divide and conquer approach. The key information required to compute the sensitivity measure in Equation (26) is the local KL divergence metric, Ω_i^{KL}, for example, associated with the ith Transformer LM layer. In our implementation, a set of prototype transformer LMs quantized using uniform precision, for example 1-bit, 2-bit, 4-bit and 8-bit are trained off-line first via ADMM optimization of Section IV. The performance sensitivity in Equation (26) can then be computed first locally for each layer by replacing the full precision parameters using each of the above four possible quantization choices at that layer only, before taking the sum to produce the combined net KL divergence metric when a particular mixed precision based local quantization configuration is applied across the entire Transformer LM.

In order to further improve the efficiency when computing the KL divergence based sensitivity measure in Equation (26), a very small number of training data samples, as little as the data of one single mini-batch (batch size set as 32 throughout this paper) can be used, and adopted in all experiments of this paper. In practice this was found to produce quantized LM perplexity comparable to that obtained by computing the KL metric over the entire training data set. An ablation study on the relationship between the amount of Switchboard training data used to compute the KL divergence metric and determine the resulting mixed precision for Transformer LM quantization and the resulting LM's perplexity performance is shown in Table I. In order to further ensure efficiency, the resulting mixed precision quantized LMs using varying automatically learned local precision settings together with quantized parameters inherited from uniform precision models of different bit-widths are fine-tuned, rather than retrained from scratch. In practice, this was found to produce performance comparable to re-training them from scratch after determining the precision settings, as is illustrated

LM	quant. estim.	quant. method	#bit	PPL		
				1	50%	100%
Transformer	-	-	32	-	-	40.7
	ADMM	KL	1.9	46.0	46.3	45.9
		Hes	1.9	46.4	46.0	46.2

in the example contrast of Table II for 2-bit and 4-bit KL mixed precision quantized Transformer LMs on the Switchboard data.

An example application of KL divergence based mixed precision quantization of Transformer and LSTM-RNN LMs are shown in Figs. 1 and 2 respectively.

B. Curvature Based Mixed Precision Quantization

The second approach to measure performance sensitivity to quantization examines the local training data log-likelihood curvature. Under mild assumptions such that the parameters of a DNN is twice differentiable and have converged to a local optimum, it has been shown in previous researches [35] that the performance sensitivity to quantization, when using a given precision setting, can be expressed as the squared quantization error further weighted by the parameter Hessian matrix trace. For any quantization $f(\cdot)$ being applied to the parameters $\boldsymbol{\Theta}$ of a L layer Transformer LM, the total performance sensitivity is given by the following sum of Hessian trace weighted squared quantization error.

$$\Omega^{\text{Hes}} = \sum_{i=1}^{L} \Omega_i^{\text{Hes}} = \sum_{i=1}^{L} Tr(\boldsymbol{H}_i) \cdot ||f(\boldsymbol{\Theta}_i) - \boldsymbol{\Theta}_i||_2^2 \quad (27)$$

Intuitively for each cluster of weight parameters (for example of the same layer) to be quantized using a particular precision bit width, given the same amount of model parameter changes resulted from quantization, a smaller Hessian matrix trace indicates a lower performance sensitivity to quantization.

Given a target average quantization precision such as 2-bit, the local quantization setting, for example, used in each Transformer LM layer, should be selected such that the total performance sensitivity in Equation (27) is minimized while satisfying the target model size constraint. Similar to the KL divergence based mixed precision quantization, a set of prototype Transformer or LSTM-RNN LMs quantized using uniform precision, for example 1-bit, 2-bit, 4-bit and 8-bit, are trained off-line first using the ADMM optimization in Section IV. The log-likelihood curvature based performance sensitivity in Equation (27) can then be computed locally for each layer using each quantization choice before taking the sum to produce the combined net sensitivity measure for any mixed precision setting being considered. For larger LSTM-RNN or Transformer LMs containing up to hundreds of millions of parameters, and large DNNs in general, directly computing the Hessian matrix and its trace is computationally infeasible. In order to address this issue, an efficient

TABLE II
PERFORMACE CONTRAST BETWEEN 2-BIT AND 4-BIT KL MIXED PRECISION QUANTIZED TRANSFORMER LMS ON THE SWITCHBOARD DATA CONSTRUCTED USING EITHER POST PRECISION LEARNING MODEL FINE-TUNING (LM 2, 4), OR RETRAINING FROM SCRATCH (LM 1, 3)

| ID | LM | quant. method | train method | #bit | PPL | eval2000 | | rt02 | | | rt03 | | WER |
						swbd	callhm	swbd1	swbd2	swbd3	fsh	swbd	avg.
1	Transformer	KL	re-train from scratch	1.9	46.2	7.2	13.4	8.4	10.6	14.0	8.3	14.3	10.9
2			post-train fine-tuning		45.9	7.2	13.4	8.3	10.7	14.0	8.3	14.3	10.9
3			re-train from scratch	4	43.8	7.1	13.3	8.2	10.5	14.0	8.3	14.3	10.8
4			post-train fine-tuning		44.0	7.1	13.3	8.2	10.5	14.0	8.3	14.2	10.8

stochastic linear algebra approach based on the Huchinson's Algorithm [89] is used to approximate the Hessian trace,

$$Tr(\boldsymbol{H}) \approx \frac{1}{m} \sum_{i=1}^{m} \boldsymbol{z}_i^\top \boldsymbol{H} \boldsymbol{z}_i \tag{28}$$

where the expensive matrix multiplication between $\boldsymbol{H}$ and $\boldsymbol{z}_i$ can be avoided, and efficiently computed using Hessian-free approaches [35]. $\boldsymbol{z}_i$ is a random vector sampled from a Gaussian Distribution $\mathcal{N}(\boldsymbol{0}, \boldsymbol{1})$. Following the previous research reported in [37], the maximum number of Hutchinson steps set as $m = 50$ is found sufficient to obtain an accurate Hessian approximation for computing the curvature based quantization sensitivity of Equation (27), and used throughout the experiments of this paper. Again for efficiency, a small subset of the training data from a randomly drawn mini-batch (batch size 32) is used, in common with the previous KL divergence metric. Further analysis on the relationship between the sampled training data size and the resulting quantized Transformer LMs perplexity performance is shown in the last line of Table I for the Switchboard data. An example application of the above curvature performance sensitivity based mixed precision quantization of a Transformer LM is shown in Fig. 3.

C. Architecture Search Based Mixed Precision Quantization

The third solution to automatically learn the optimal local quantization precision settings is to use mixed precision based neural architecture search (NAS) approaches. Neural architecture search (NAS) techniques [90] can efficiently automate neural network structure designs that have been largely based on expert knowledge or empirical choice to date. Among existing NAS methods, differentiable neural architecture search (DARTS) [69]–[71], [91], [92] benefits from a distinct advantage of being able to simultaneously compare a very large number of candidate architectures during search time. This is contrast to earlier and more expensive forms of NAS techniques based on, for example, genetic algorithms [72] and Reinforcement learning (RL) [65], [93], where explicit system training and evaluation are required for a large number of candidate structures under consideration.

Neural architecture search using DARTS is performed over an over-parameterized super-network containing paths connecting all candidate DNN structures to be considered. The search is transformed into the estimation of the weights assigned to each candidate neural architecture within the super-network. The optimal architecture is obtained by pruning lower weighted paths. This allows both architecture selection and candidate
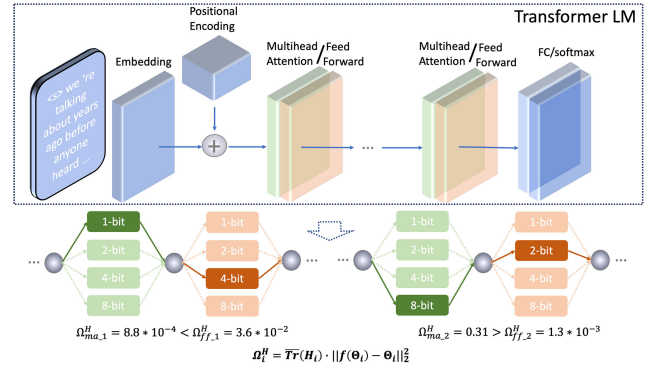

Fig. 1. An example of mixed precision quantization of a Transformer LM using KL-divergence based mixed precision quantization. For the first Transformer module positioned right after the embedding and position encoding layer, its multi-head attention layer (green) uses 2-bit quantization while its feed forward layer (orange) uses binary quantization precision, as determined by the KL-divergence based sensitivity measure.

DNN parameters to be consistently optimized within the same super-network model.

The key difference from conventional NAS tasks is that instead of selecting over heterogeneous neural building structures, for example, varying hidden layer left and right context offsets or projection layer dimensionality in LF-MMI trained time delay neural networks (TDNNs) [91], different quantized neural building blocks, for example, LSTM or Transformer LM modules of different bit-widths are considered. This crucial difference requires the associated mixed precision quantization super-network to be specially designed. Such super-network is constructed by first separately training transformer LMs using uniform precision, for example 1-bit, 2-bit, 4-bit and 8-bit, using ADMM optimization, before connecting these uniform precision quantized Transformer LMs at each layer, where the system specific activation outputs are linearly combined using a set of quantization precision selection weights as in the following equation:

$$\boldsymbol{o}^l = \sum_{n \in N} \frac{exp(a_n^l)}{\sum exp(a_n^l)} F(f_n(\boldsymbol{\Theta}^l), \boldsymbol{o}^{l-1}) \tag{29}$$

where a_n^l is the architecture weights using n-bit quantization for the l-th cluster of weight parameters. $f_n(\cdot)$ represents n-bit quantization of the weight parameter $\boldsymbol{\Theta}^l$ and $F(\cdot)$ is the underlying layer's activation function. The set of possible quantization precision setting, $N = \{1, 2, 4, 8\}$, is used throughout this paper. An example of such mixed precision Transformer super-network is shown in Fig. 4.
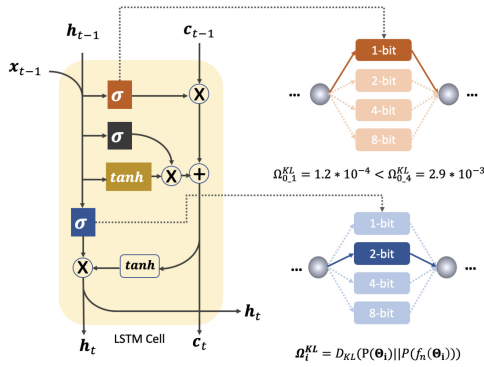
Fig. 2.   An example of auto-configured mixed precision quantization of a 2-layer LSTM-RNN LM with its forget, input and output Sigmoid gates σ marked in orange, grey and dark blue, and cell gate $\tanh$ in brown respectively. The forget gate (orange) parameters use 1-bit quantization, while those of the output gate (dark blue) use 2-bit quantization, as determined by the KL-divergence based sensitivity measure.
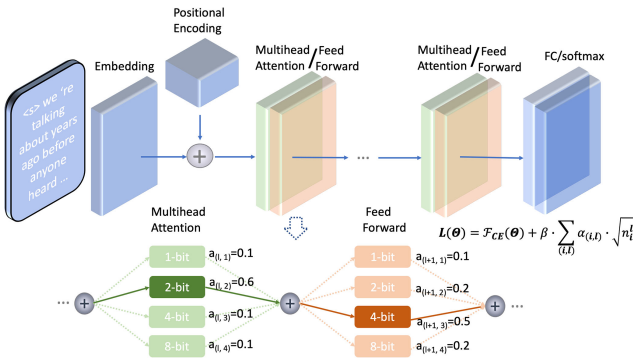

Fig. 3.   An example of auto-configured mixed precision quantization of a Transformer LM using curvature based sensitivity measure. For the first Transformer module positioned right after the embedding and position encoding layer, its multi-head attention layer (green) uses binary quantization while its feed forward layer (orange) uses 4-bit quantization precision, as determined by the Hessian-trace weighted quantization sensitivity measure.

In order to avoid the trivial selection of the longest, most generous quantization bit width, these precision selection weights learning can be further constrained by a model complexity penalty term with respect to the number of bits retained after quantization, in order to obtain a target average quantization precision, for example, 2-bit,

$$\Omega^{NAS} = \mathcal{F}_{ce}(\boldsymbol{\Theta}) + \beta \sum_{(n,l)} a_n^l \cdot \sqrt{n} \qquad (30)$$

where $\mathcal{F}_{ce}(\boldsymbol{\Theta})$ is the standard cross-entropy loss.

VI. Experiments

In this section the performance of mixed precision quantized LSTM-RNN and Transformer LMs are evaluated on two speech recognition systems both of which use state-of-the-art LF-MMI sequence trained hybrid time delay neural networks (TDNNs) acoustic models with factored weights and additional convolutional layers [78]. Speech perturbation based data augmentation and i-Vector based speaker adaptation are also employed. Modified KN smoothed 4-gram back-off LMs are used during the initial N-best list generation pass before various NNLMs are then
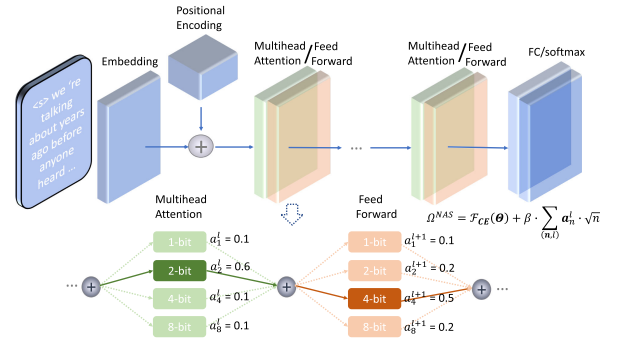

Fig. 4.   An example of auto-configured mixed precision quantization of a Transformer LM using mixed precision architecture search. For the first Transformer module, its multi-head attention layer is uses 2-bit quantization (green) given the associated selection weight of 0.6 while its feed forward layer uses 4-bit quantization precision (orange) given the associated selection weight of 0.5, as the 1-best choice selected from the mixed precision NAS super-network.

applied in the following rescoring stage. In Section VI-A, the first set of experiments conducted on the Switchboard I (SWBD) corpus [81] are presented. In Section VI-B another comparable set of experiments are carried out on the AMI meeting room data [82].

All the LSTM-RMM LMs investigated in this paper consist of 2 LSTM layers while both the input word embedding and hidden layer sizes were set as 1024. All the Transformer LMs used in this paper contain 6 Transformer layers. The dimensionality of all query, key and value embedding and hidden vectors are set as 512 for each batch of data. All the mixed precision quantized LSTM-RNN LMs and Transformer LMs of this paper use layer or node level precision settings that are set either manually as equal bit-widths (1-bit, 2-bit, 4-bit or 8-bit), or automatically learned using the KL, curvature or NAS based mixed precision quantization methods of Sections V. All NNLMs were tained using a single NVIDIA Tesla V100 Volta GPU card. Statistical significance test was conducted at level $\alpha = 0.05$ based on matched pairs sentence segment word error (MAPSSWE) for recognition performance analysis. This is used to identify various "lossless" neural LM quantization configurations that incur no recognition performance degradation, or statistically significant word error rate (WER) increase.

The implementation used to evaluate the mixed precision quantization methods of this paper is exclusively based on the existing low-bit quantized precisions that are already natively supported by the NVidia Tesla V100 GPU. These include the use of the Boolean and masking operators to implement 1-bit quantization, and the INT8 data type used to implement 2, 4 and 8-bit quantization. In case of 2-bit and 4-bit quantization, extra padded bits of zero were also included.

A. Experiments on Conversational Telephone Speech

The Switchboard I telephone speech corpus we use consists of approximately 300 hours of audio data released by LDC (LDC97S62). Following the Kaldi toolkit [95] and its CNN-TDNN recipe,[1] LF-MMI trained CNN-TDNN acoustic models [78] with data augmentation and i-Vector adaptation [79]

---

[1]egs/swbd/s5c/local/chain/run_cnn_tdnn_1a.sh

TABLE III
PERFORMANCE OF THE BASELINE FULL PRECISION (LM 1), UNIFORM PRECISION QUANTIZED (LM 2-11) AND MIXED PRECISION QUANTIZED RNNLMs WITH
LOCAL LAYER OR NODE LEVEL PRECISION SET EITHER MANUALLY WITH EQUAL BIT-WIDTH (LM 12-15, 16-19) OR AUTOMATICALLY LEARNED (LM 20-25)
USING KL, CURVATURE (HES) OR NAS BASED MIXED PRECISION QUANTIZATION METHODS OF SECTION V ON SWITCHBOARD NIST HUB5'00, RT02 AND RT03.
EQUAL WEIGHT INTERPOLATION OF N-GRAM AND NEURAL LMs USED IN N-BEST RESTORING. ALL WER CHANGES OF NO STATISTICAL SIGNIFICANCE
(MAPSSWE, $\alpha = 0.05$) OVER THE FULL PRECISION BASELINE (LM 1) ARE MARKED WITH "∗". OFFLINE, POST-TRAINING QUANTIZED LMs' PERFORMANCE
ARE ALSO SHOWN (LM 6-9)

ID	quant. prec.	param. estim.	quant. method	#bit	PPL	eval2000		rt02			rt03		WER(%) avg.	model size(MB)	comp. ratio	eval. time (ms./word)
						swbd	callhm	swbd1	swbd2	swbd3	fsh	swbd				
0		4-gram		-	85.3	9.5	17.8	11.4	15.1	19.5	12.4	19.3	15.0	-	-	-
1		baseline (full precision)		32	40.7	7.1	13.2	8.2	10.7	13.8	8.4	14.1	10.8	183	-	0.226
2	uniform prec.	BP [31], [32] (modified)	manual define	1	52.4	8.3	15.9	9.6	12.0	15.8	9.2	15.6	12.3	6.1	30.0	0.086
3				2	50.1	8.0	15.1	9.3	11.6	15.4	9.1	15.4	12.0	11.8	15.5	0.140
4				4	49.7	7.8	14.5	8.7	11.3	15.1	8.9	15.0	11.6	23.5	7.9	0.142
5				8	44.2	7.4	13.9	8.6	11.1	14.6	8.7	14.8	11.3	46.2	3.9	0.141
6		Post-training [94] Offline Quant		1	50.3	8.2	15.4	9.6	12.0	15.7	9.1	15.6	12.2	6.1	30.0	0.085
7				2	47.6	7.9	14.6	9.2	11.3	15.2	9.0	15.2	11.8	11.8	15.5	0.141
8				4	47.1	7.8	14.3	8.7	11.2	14.9	8.8	15.0	11.5	23.5	7.9	0.142
9				8	44.8	7.3	13.6	8.4	11.1	14.4	8.7	14.6	11.2	46.2	3.9	0.141
10		ADMM (global)		1	49.9	8.0	14.8	9.6	11.8	15.7	9.1	15.4	12.1	6.1	30.0	0.083
11				8	43.0	7.2	13.6	8.4	11.0	14.3	8.6	14.6	11.1	46.2	3.9	0.144
12	mixed prec.	ADMM (multiple eql. #bit node)		1	49.4	7.8	14.2	9.4	11.7	15.5	8.8	15.1	11.8	7.6	24.1	0.102
13				2	45.3	7.6	14.0	9.1	11.4	15.2	8.7	14.9	11.6	13.3	13.6	0.155
14				4	44.7	7.3*	13.5*	8.6	11.0	14.5	8.6*	14.7	11.2	24.8	7.4	0.152
15				8	42.1	7.2*	13.3*	8.3*	10.9*	14.0*	8.4*	14.3*	10.9*	47.5	3.8	0.158
16		ADMM (multiple eql. #bit layer)		1	50.3	8.0	15.3	9.5	11.8	15.6	9.1	15.4	12.1	6.1	30.0	0.088
17				2	48.2	7.7	14.5	9.2	11.4	15.3	8.9	15.2	11.7	11.8	15.5	0.148
18				4	46.3	7.6	14.0	8.6	11.2	14.7	8.8	14.9	11.4	23.5	7.9	0.148
19				8	43.9	7.3*	13.5*	8.4*	11.0*	14.3	8.6*	14.5	11.1*	46.2	3.9	0.147
20		ADMM (layer level variable)	Hes	1.9	46.2	7.5	14.1	8.8	11.1	14.5	8.9	15.0	11.4	11.7	15.6	0.130
21				4	43.1	7.4	13.6	8.4	10.9	14.1	8.6	14.5	11.0	23.8	7.7	0.140
22			KL	1.9	45.9	7.4*	13.9	8.5*	10.9*	14.2	8.7*	14.8	11.2	11.7	15.6	0.132
23				4	41.9	7.3	13.4	8.3	10.8	14.0	8.5	14.4	10.9	23.7	7.7	0.139
24			NAS	2.3	48.3	7.7	14.3	9.0	11.4	14.9	9.0	15.2	11.6	13.1	13.9	0.136
25				4	46.8	7.6	14.1	8.5	11.2	14.6	8.7	14.7	11.3	23.4	7.8	0.141

were then built. The CNN-TDNN network consisted of 6 convolutional layers followed by 8 context-splicing TDNN layers with 1536 nodes per layer. A 160-dimensional factored linear projection was employed prior to affine transformation in each context-splicing layer other than the first one. ReLU activation functions were used, followed by batch normalization and dropout operations. Left and right context offsets of $\{-3, 3\}$ were also used in the context-splicing layers. More detailed description fo the baseline system's acoustic modelling configuration can be found in [96]. The Switchboard NIST Hub5'00, RT02 and RT03 evaluation sets were used. A 30 K word recognition lexicon was used. Various LSTM and Transformer LMs trained on the Switchboard and Fisher transcripts (LDC2004T19, LDC2005T19) were used to rescore the 4-gram LM produced N-best lists ($N = 20$).

1) Experiments on LSTM-RNN LMs using baseline full precision, various uniform and mixed quantization settings are shown in Table III. All WER changes as the result of quantization that are of no statistical significance (MAPSSWE, $\alpha = 0.05$) are marked by "∗". Several trends can be found here:

- Among the uniform precision quantized LMs, the LSTM-RNN LMs of 1-bit and 8-bit precisions trained using the ADMM optimization of Section IV consistently outperform those of comparable bit-widths but built using the modified BP algorithm (LM 10 and 11 vs. LM 2 and 5 in Table III). In particular, for 1-bit quantization, the ADMM optimization method produced WER reductions up to 1.1% absolute on the **callhm** data of the Hub5'00 set over the traditional modified BP algorithm (LM 10 vs. LM 2 in Table III). The advantage of the ADMM algorithm in terms of convergence speed and training efficiency is further illustrated in left part of Fig. 5.

- The mixed precision quantized LSTM LMs (LM 12-25 in Table III) consistently outperform the uniform precision quantized models (LM 2-11). For example, given the same quantization precision at approximately 2-bit (compression ratio of 16 times over 32-bit full precision), a wide range of mixed precision quantized LMs (LM 13, 17, 20, 22, 24 in Table III) produced statistically significant 0.3-0.8% average WER reduction across three test data sets over with the comparable 2-bit uniform precision quantization baseline (LM 3).

- Note that the first two of these mixed precision quantized LSTM LMs (LM 13, 17 in Table III) used multiple layer or node level locally applied quantization tables of the same bit-width manually set as 2-bit. As expected, a finer modelling granularity provided by the node level local quantization (LM 17) marginally outperformed the use of layer level (LM 13) by 0.1% on average in WER, albeit with a smaller compression ratio of 13.6 vs. 15.5. Similar trends can be found when increasing the quantization precision from 2-bit to 8-bit.[2]

- Among the three mixed precision quantization LSTM LMs with approximately 2-bit precision (LM 20, 22, 24 in Table III), the KL quantized LM (LM 22) with a 1.9-bit average precision gives the lowest average WER of 11.2%. These three mixed precision LMs' respective local selection of quantization bit-widths at different gates of various LSTM layers are shown in Fig. 6(a)-6(c). Note that KL and curvature based quantization (LM 20, 22 in Table III) both

---

[2]Considering the trade-off between WER and compression ratio for node and layer level local quantization precision settings, the subsequent experiments will focus on using layer level mixed precision quantization.
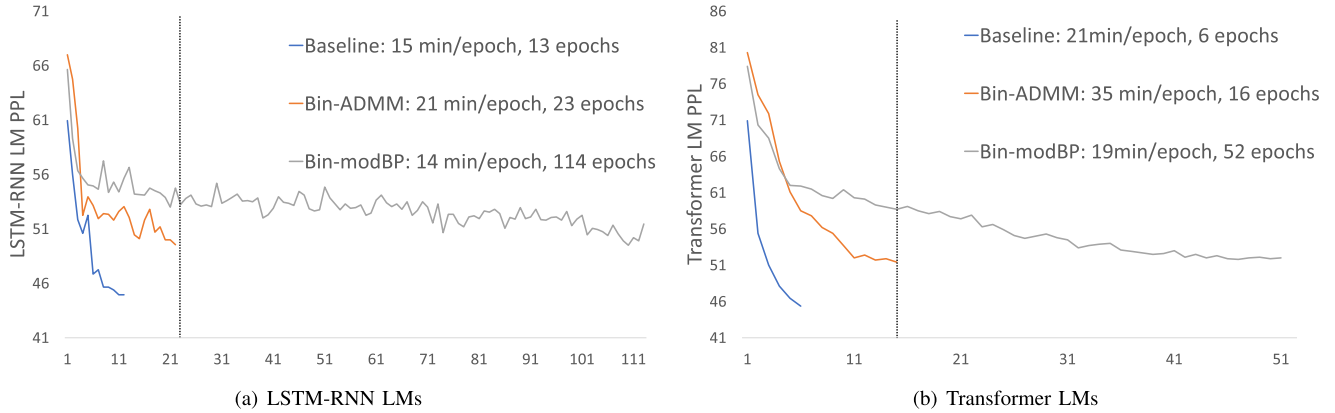
(a) LSTM-RNN LMs

(b) Transformer LMs

Fig. 5.   Perplexity and convergence speed comparison on LSTM-RNN LMs (figure (a), showing LM 1, 2, 6 in Table III) and Transformer LMs (figure (b), showing LM 1,2,6 in Table IV) between baseline full precision models (*(Baseline)*, binarized models trained using modified back-propagation (*Bin-modBP*) [31], [32] and ADMM trained binarized models (*Bin-ADMM*) on Switchboard validation data.

(a) KL quantized LSTM LM (avg. 2-bit)

(b) Curvature quantized LSTM LM (avg. 2-bit)

(c) NAS quantized LSTM LM (avg. 2-bit)

(d) KL quantized Transformer LM (avg. 2-bit)

(e) Curvature quantized Transformer LM (avg. 2-bit)

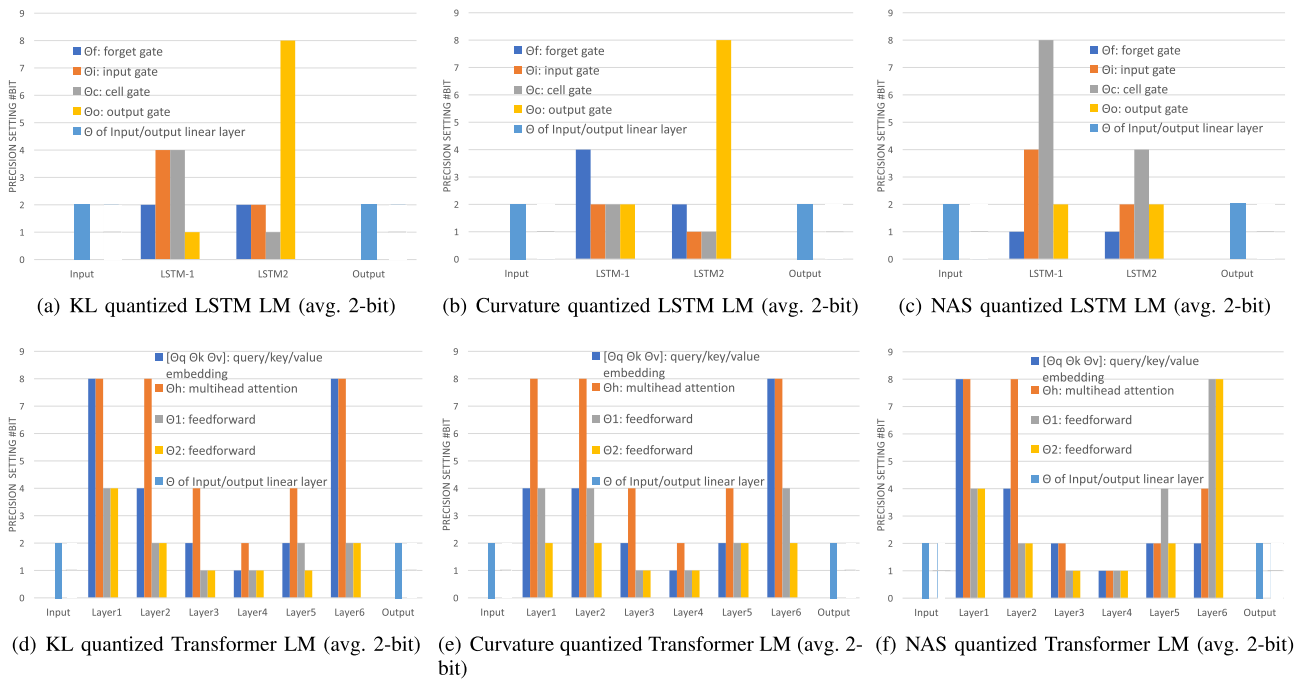(f) NAS quantized Transformer LM (avg. 2-bit)

Fig. 6.   Number of bits used in local quantization of SWBD data trained neural LMs automatically derived using KL, curvature and NAS based mixed precision quantization of Section V for: 1) LSTM-RNN LMs ((a)-(c), also shown as LM 16-18 in Table III) at their input/output linear layers (light blue), and individual forget (dark blue), input (orange), cell (grey) and output (yellow) gates inside LSTM layers; 2) Transformer LMs ((d)-(f), also shown as LM 16-18 in Table IV) at their input/output linear layers (light blue), and 1st/2nd multihead attention sublayers (dark blue/orange), following 1st/2nd feedforward sublayers (grey/yellow) within each of 6 Transformer layers.

selected 8-bit precision for the 2nd LSTM layer's output gate parameters, indicating a larger performance sensitivity to quantization measured at the LSTM cells' output gate.
- Among all the quantized LSTM LMs marked with "∗" in Table III that incur no statistically significant WER increase against the full precision baseline (LM 1), the 1.9-bit KL mixed precision quantized model (LM 22) also produced the largest "lossless" compression ratio of 15.6.

2) Experiments on Transformer LMs designed using a comparable set of contrasts previously used in Table III for LSTM-RNN LMs are shown in Table IV, where a large number of quantized

Transformer LMs featuring different forms of mixed quantization schemes (LM 12 to 25 in Table IV, marked with "∗") produced lossless compression with no statistically significant WER changes relative to the baseline full precision model (LM 1). Several other trends that are similar to those found in Table III on LSTM-RNN LMs are summarized below:
- Among the uniform precision quantized LMs, the Transformer LMs of 1-bit and 8-bit precision trained using ADMM optimization of Section IV marginally outperform, or are comparable in performance to those two of same precisions using the modified BP algorithm (LM 10 and

TABLE IV
PERFORMANCE OF THE BASELINE FULL PRECISION (LM 1), UNIFORM PRECISION QUANTIZED (LM 2-11) AND MIXED PRECISION QUANTIZED TRANSFORMER LMS WITH LOCAL LAYER OR NODE LEVEL PRECISION SET EITHER MANUALLY WITH EQUAL BIT-WIDTH (LM 12-15, 16-19) OR AUTOMATICALLY LEARNED (LM 20-25) USING KL, CURVATURE (HES) OR NAS BASED MIXED PRECISION QUANTIZATION METHODS OF SECTION V ON SWBD NIST HUB5'00, RT02 AND RT03. EQUAL WEIGHT INTERPOLATION OF N-GRAM AND NEURAL LMS USED IN N-BEST RESTORING. ALL WER CHANGES OF NO STATISTICAL SIGNIFICANCE (MAPSSWE, $\alpha = 0.05$) OVER THE FULL PRECISION BASELINE (LM 1) ARE MARKED WITH "∗". OFFLINE, POST-TRAINING QUANTIZED LMS' PERFORMANCE ARE ALSO SHOWN (LM 6-9)

ID	quant. prec.	param. estim.	quant. method	#bit	PPL	eval2000		rt02			rt03		WER(%) avg.	model size(MB)	comp. ratio	eval. time (ms./word)
						swbd	callhm	swbd1	swbd2	swbd3	fsh	swbd				
0		4-gram		-	85.3	9.5	17.8	11.4	15.1	19.5	12.4	19.3	15.0	-	-	-
1		baseline (full precision)		32	40.7	7.1	13.4	8.2	10.5	14.0	8.3	14.3	10.8	106	-	0.127
2	uniform prec.	BP [31], [32] (modified)	manual define	1	52.4	7.5	14.2	8.5	11.0	14.7	9.0	14.9	11.4	3.6	29.5	0.043
3				2	50.1	7.5	14.1	8.5	11.0	14.6	8.9	14.8	11.3	7.2	14.7	0.059
4				4	49.7	7.4*	13.9*	8.4*	10.8*	14.4*	8.7*	14.6*	11.0	13.8	7.7	0.063
5				8	44.2	7.3*	13.7*	8.4*	10.8*	14.5*	8.6*	14.5*	10.9	27.0	3.9	0.065
6		Post-training [94] Offline Quant		1	50.7	7.4	14.1	8.3	10.9	14.4	8.8	14.6	11.2	3.6	29.5	0.044
7				2	49.2	7.3	13.8	8.3	10.7	14.1	8.5	14.5	11.1	7.2	14.7	0.061
8				4	47.3	7.2	13.7	8.3	10.6	14.0	8.4	14.4	10.9	13.8	7.7	0.059
9				8	45.0	7.2	13.5	8.2	10.6	14.0	8.3	14.4	10.9	27.0	3.9	0.062
10		ADMM (global)		1	51.2	7.5	14.1	8.5	10.9	14.5	8.8	14.7	11.3	3.6	29.5	0.046
11				8	44.0	7.2	13.6	8.4	10.8	14.2	8.5	14.5	10.9	27.0	3.9	0.062
12	mixed prec.	ADMM (multiple eql. #bit node)	manual define	1	49.4	7.3*	13.8*	8.3*	10.8*	14.2*	8.6*	14.5*	11.0*	4.3	24.7	0.059
13				2	45.3	7.2*	13.5*	8.3*	10.6*	14.0*	8.4*	14.4*	10.9*	7.7	13.8	0.076
14				4	44.7	7.1*	13.5*	8.2*	10.5*	14.0*	8.3*	14.3*	10.8*	15.3	7.0	0.075
15				8	42.1	7.2*	13.5*	8.3*	10.6*	14.0*	8.3*	14.4*	10.9*	28.5	3.7	0.076
16		ADMM (multiple eql. #bit layer)		1	50.3	7.3	14.0	8.3*	10.8*	14.3*	8.7*	14.6*	11.2*	3.7	29.5	0.048
17				2	48.2	7.2*	13.6*	8.3*	10.6*	14.0*	8.5*	14.4*	11.0*	7.2	14.7	0.070
18				4	46.3	7.2*	13.6*	8.2*	10.7*	14.0*	8.3*	14.3*	10.9*	13.8	7.7	0.072
19				8	43.9	7.2*	13.5*	8.3*	10.7*	14.0*	8.3*	14.4*	10.9*	27.0	3.9	0.072
20		ADMM (layer level variable)	Hes	1.9	46.2	7.2*	13.5*	8.3*	10.7*	14.0*	8.3*	14.4*	10.9*	7.0	15.1	0.062
21				4	44.2	7.2	13.4	8.2	10.6	14.0	8.3	14.3	10.8	13.7	7.7	0.069
22			KL	1.9	45.9	7.2*	13.4*	8.3*	10.7*	14.0*	8.3*	14.3*	10.9*	7.0	15.1	0.064
23				4	44.0	7.1	13.3	8.2	10.5	14.0	8.3	14.2	10.8	13.7	7.7	0.070
24			NAS	2.3	48.3	7.3*	13.7*	8.4*	10.8*	14.0*	8.4*	14.5*	11.1*	7.5	14.2	0.065
25				4	45.8	7.2	13.5	8.2	10.6	14.1	8.3	14.3	10.9	13.4	7.9	0.068

11 vs. LM 2 and 5 in Table IV). The advantage of the ADMM based Transformer LM estimation in terms of convergence speed and training efficiency is shown in right part of Fig. 5.

- The mixed precision quantized Transformer LMs (LM 8-18 in Table IV) again consistently outperform the uniform precision quantized baselines (LM 2-11) across different lower bit-widths ranging from 1-bit, 2-bit to 4-bit. For example, given the same quantization precision at approximately 2-bit (compression ratio of 16 times), a wide range of mixed precision quantized Transformer LMs (LM 13, 20, 22 in Table IV) produced statistically significant 0.4% absolute WER reduction averaged across all data sets over with the comparable 2-bit uniform precision quantization LM (LM 3 in Table IV).

- Among the three mixed precision quantized Transformer LMs with approximately 1.9-bit precision (LM 20, 22, 24 in Table IV), the KL and curvature based mixed precision quantized Transformer LMs (LM 22) with a 1.9-bit average precision produced the lowest average WER of 10.9%, as well as the largest "lossless" compression ratio of 15.1 among all the quantized Transformer LMs marked with "∗" in Table IV that incur no statistically significant WER increase over the full precision baseline (LM 1 in Table IV). Their respective local selection of quantization bit-widths at different sublayers within various Transformer layers are shown in Fig. 6(e)-6(f), where an expected general trend is found that both the lower Transformer layers heavily tasked with de-noising the data, and the top Transformer layer used to immediately predict detailed word probabilities over a large vocabulary, require longer quantization precision settings (2-bit to 8-bit) than those middle Transformer layers (1-bit to 4-bit).

- It is also worth noting the 4-bit ADMM node level locally quantized Transformer LM (LM 14 in Table IV), and the 4-bit KL or curvature mixed precision quantized Transformer LM (LM 21, 23 in Table IV), both produced no WER degradation against the full precision model (LM 1), albeit with a smaller compression ratio of 7.0.

B. Experiments on AMI Meeting Room Data

The Augmented Multi-party Interaction (AMI) speech corpus consists of approximately 100 hours of audio data collected using both headset microphone and distant microphone arrays from the meeting environment. Following the Kaldi recipe,[3] three LF-MMI trained [78] acoustic models with speech perturbation based data augmentation and i-Vector based speaker adaptation [79] were then constructed. The AMI 8.9-hour **dev** and 8.7-hour **eval** sets recorded under close talking microphone (**ihm**), single distant microphone (**sdm**) and multiple distant microphones (**mdm**) were used. A 47 K word recognition lexicon was used. Various LSTM and Transformer LMs based on the same configurations as those on the Switchboard data in Table III and IV were trained using a mixture of text sources of 15 M words including Fisher transcripts and 3 times of AMI transcriptions before being used to rescore the 3-gram LM produced N-best lists ($N = 20$). All other experimental configurations remain the same as the Switchboard experiments of Section VI-A.

---

[3]egs/ami/s5c/local/chain/run_tdnn.sh

TABLE V
PERFORMANCE OF THE BASELINE FULL PRECISION (LM 1), UNIFORM PRECISION QUANTIZED (LM 2-7) AND MIXED PRECISION QUANTIZED RNNLMs WITH LOCAL LAYER OR NODE LEVEL PRECISION SET EITHER MANUALLY WITH EQUAL BIT-WIDTH (LM 8-11, 12-15) OR AUTOMATICALLY LEARNED (LM 16-21) USING KL, HES OR NAS BASED MIXED PRECISION QUANTIZATION METHODS OF SECTION V ON AMI DEV AND EVAL SETS OF IHM, MDM8 AND SDM1 CONDITIONS. EQUAL WEIGHT INTERPOLATION OF N-GRAM AND NEURAL LMs USED IN N-BEST RESTORING. ALL WER CHANGES OF NO STATISTICAL SIGNIFICANCE (MAPSSWE, $\alpha = 0.05$) OVER THE FULL PRECISION BASELINE (LM 1) ARE MARKED WITH "*"

ID	param. prec.	quant. estim.	quant. method	#bit	PPL	ihm		mdm8		sdm1		model size(MB)	comp. ratio	eval. time (ms./word)
						dev	eval	dev	eval	dev	eval			
0	4-gram			-	93.6	18.1	18.4	33.5	35.3	34.2	40.5	-	-	-
1	baseline (full precision)			32	57.6	16.4	16.2	27.7	30.8	29.9	34.7	248	-	0.290
2	uniform prec.	BP [31], [32] (modified)		1	78.6	17.2	16.9	32.5	34.1	33.9	40.1	8.2	30.2	0.108
3				2	67.9	16.8	16.6	30.1	33.2	31.6	37.9	16.2	15.3	0.177
4				4	62.6	16.7	16.5	29.3	32.1	30.7	35.7	31.5	7.9	0.179
5				8	60.3	16.5*	16.3*	28.6	31.8	30.4	35.3	62.8	3.9	0.175
6		ADMM (global)		1	74.2	17.0	16.8	31.2	33.7	32.9	39.6	8.2	30.2	0.110
7				8	59.3	16.6*	16.3*	28.5	31.6	30.4	35.2	62.8	3.9	0.176
8	mixed prec.	ADMM (multiple eql. #bit node)	manual define	1	68.2	16.9	16.7	30.0	32.6	32.4	38.6	9.5	26.1	0.145
9				2	61.9	16.6*	16.5*	29.1	31.9	30.9	35.9	17.6	14.1	0.205
10				4	58.5	16.4*	16.3*	28.6	31.2	30.5	35.2	33.0	7.5	0.207
11				8	57.9	16.4*	16.3*	28.4	31.1*	30.1*	34.9*	64.5	3.8	0.204
12		ADMM (multiple eql. #bit layer)		1	70.3	17.0	16.8	30.9	33.2	32.7	39.1	8.4	30.1	0.116
13				2	63.1	16.7	16.6	29.4	32.1	31.2	36.2	16.2	15.3	0.189
14				4	59.2	16.6*	16.4*	28.9	31.7	30.9	35.4	31.5	7.9	0.190
15				8	58.4	16.5*	16.3*	28.5	31.4	30.4	35.2	62.8	3.9	0.192
16		ADMM (layer level variable)	Hes	1.9	60.7	16.5*	16.4*	28.6	31.8	30.8	35.3	16.0	15.5	0.165
17				4.0	59.0	16.5*	16.4*	28.5	31.2	30.3*	35.1	31.3	7.9	0.185
18			KL	1.8	59.2	16.5*	16.3*	28.5	31.6	30.6	35.2	15.8	15.7	0.167
19				3.8	58.8	16.5*	16.3*	28.4	31.3	30.2*	35.0*	30.8	8.1	0.183
20			NAS	2.0	63.6	16.8	16.5	28.9	32.0	31.1	35.9	16.2	15.3	0.171
21				3.8	61.7	16.5*	16.4*	28.8	31.6	30.7	35.5	30.8	8.1	0.186

TABLE VI
PERFORMANCE OF THE BASELINE FULL PRECISION (LM 1), UNIFORM PRECISION QUANTIZED (LM 2-7) AND MIXED PRECISION QUANTIZED TRANSFORMER LMs WITH LOCAL LAYER OR NODE LEVEL PRECISION SET EITHER MANUALLY WITH EQUAL BIT-WIDTH (LM 8-11, 12-15) OR AUTOMATICALLY LEARNED (LM 16-21) USING KL, HES OR NAS BASED MIXED PRECISION QUANTIZATION METHODS OF SECTION V ON AMI DEV, EVAL SETS OF IHM, MDM8 AND SDM1 CONDITIONS. EQUAL WEIGHT INTERPOLATION OF N-GRAM AND NEURAL LMs USED IN N-BEST RESTORING. ALL WER CHANGES OF NO STATISTICAL SIGNIFICANCE (MAPSSWE, $\alpha = 0.05$) OVER THE FULL PRECISION BASELINE (LM 1) ARE MARKED WITH "*"

ID	quant. prec.	quant. estim.	quant. method	#bit	PPL	ihm		mdm8		sdm1		model size(MB)	comp. ratio	eval. time (ms.word)
						dev	eval	dev	eval	dev	eval			
0	4-gram			-	93.6	18.1	18.4	33.5	35.3	34.2	40.5	-	-	-
1	baseline (full precision)			32	46.9	16.1	15.8	27.5	30.6	29.8	34.5	138	-	0.172
2	uniform prec.	BP [31], [32] (modified)		1	64.3	16.7	16.5	29.8	33.0	31.3	37.2	4.7	29.4	0.058
3				2	58.4	16.4	16.3	28.7	31.2	30.4	35.3	9.0	15.3	0.082
4				4	54.1	16.3*	16.1*	28.3	31.0	30.3	35.1	17.9	7.7	0.080
5				8	52.7	16.2*	16.0*	28.1	30.8	30.1	34.8	35.0	3.9	0.083
6		ADMM (global)		1	59.9	16.6	16.5	29.7	32.6	31.1	37.0	4.7	29.5	0.062
7				8	49.7	16.2*	16.0*	28.0	30.7*	30.0*	34.8*	35.0	3.9	0.086
8	mixed prec.	ADMM (multiple eql. #bit node)	manual define	1	55.3	16.5	16.4	29.2	31.9	30.8	36.4	5.9	23.4	0.070
9				2	48.8	16.3*	16.2*	28.4	30.9	30.4	35.0	10.3	13.4	0.098
10				4	49.0	16.1*	15.9*	27.9	30.7	30.0*	34.7*	19.4	7.1	0.098
11				8	48.3	16.1*	15.8*	27.8*	30.7*	29.9*	34.6*	36.3	3.8	0.096
12		ADMM (multiple eql. #bit layer)		1	57.4	16.6	16.5	29.4	32.1	31.0	36.5	4.8	28.8	0.066
13				2	52.9	16.3	16.3	28.5	31.1	30.4	35.1	9.1	15.2	0.092
14				4	49.4	16.1*	16.0*	27.8*	30.8*	30.0*	34.6*	17.9	7.7	0.093
15				8	48.3	16.1*	15.9*	27.9	30.7	30.0*	34.7*	35.0	3.9	0.092
16		ADMM (layer level variable)	Hes	1.8	48.1	16.1*	16.0*	28.0	30.8	30.1*	34.7*	8.7	15.9	0.075
17				4.0	49.3	16.1*	15.9*	27.8*	30.7*	29.9*	34.6*	17.8	7.8	0.086
18			KL	1.8	47.6	16.2*	16.1*	28.1	30.9	30.1*	34.8*	8.7	15.9	0.074
19				4.0	48.9	16.1*	15.8*	27.6*	30.7*	29.9*	34.6*	17.8	7.8	0.085
20			NAS	1.9	49.8	16.3*	16.2*	28.3	31.0	30.3	35.0	9.0	15.3	0.077
21				3.9	49.7	16.2*	16.1*	28.1	30.7*	30.0*	34.9	17.7	7.8	0.087

The mixed precision quantization experiments for RNNLMs and Transformer LMs on AMI meeting room data are shown in Tables V and VI. The following trends similar to those previously found on the Switchboard data are again observed in Table V and VI for LSTM-RNN and Transformer LMs.

- The mixed precision quantized LSTM and Transformer LMs (LM 8-21 in Table V and Table VI) consistently outperform the uniform precision quantized models (LM 2-7 in Table V and Table VI). For example, given the same quantization precision at approximately 4-bit (compression ratio of 8 times over 32-bit full precision), a wide range of mixed precision quantized LMs (LM 10, 14, 17, 19, 21 in Table V and Table VI) produced statistically significant WER reductions up to 0.9% absolute (LM 19 vs. LM 4

on **mdm dev** in Table V) for LSTM LMs, and up to 0.7% absolute WER reduction (LM 19 vs. LM 4 on **mdm dev** in Table VI) for Transformer LMs over the comparable 4-bit uniform precision quantization (LM 4).

- Among the six mixed precision quantization LSTM and Transformer LMs with approximately 4-bit precision (LM 17, 19, 21 in Table V and Table VI), both the KL quantized LSTM and Transformer LMs (LM 19 in Table V and Table VI) with their corresponding 3.8-bit and 4.0-bit average precision give the lowest WERs. Both the KL and curvature based mixed precision quantized Transformer LMs (LM 17, 19 in Table VI) with 4-bit average precision produced no statistically significant recognition error rate increase relative to the full precision Transformer LM (LM 1 in Table VI) across all test sets.

VII. Conclusion

This paper presents a set of novel mixed precision based neural network LM quantization techniques for LSTM-RNNs and Transformers. In order to account for the locally varying performance sensitivity to low-bit quantization, the optimal local precision settings are automatically learned by either minimizing the KL-divergence or log-likelihood curvature based performance sensitivity measures, or derived using mixed precision neural architecture search. Quantized LSTM-RNN and Transformer LM parameters are estimated efficiently using alternating direction methods of multipliers based optimization, to address the low convergence speed issue when directly applying gradient descent methods to estimate discrete quantized neural network parameters. Experimental results conducted on two state-of-the-art speech recognition tasks suggest the proposed mixed precision neural network LM quantization methods outperform traditional uniform precision based quantization approaches, and can produce "lossless" quantization and large model size compression ratios of up to around 16 times over the full precision LSTM-RNN and Transformer LM baselines while incurring no statistically significant recognition accuracy degradation.

All of the three automatic mixed precision configuration approaches proposed in this paper aim to minimize the performance sensitivity to quantization errors. To the best of our knowledge, they are the first research work to apply mixed precision quantization to neural language models for speech recognition tasks. For all the three approaches such performance sensitivity measure is related to the empirical training data log-likelihood degradation given a particular quantization precision bit-width setting relative to the full precision model. The precise manner with which such measure is computed differs among the three. This leads to the measures of information loss via KL-divergence, rate of log-likelihood degradation using curvature, and penalized log-likelihood based mixed precision architecture search approaches respectively. Experimental results obtained both the Switchboard and AMI tasks shown in Tables II to V on LSTM-RNN and Transformer LMs suggest the performance between the KL-divergence and log-likelihood curvature methods is consistently close. This is expected as both are measuring

similar forms of log-likelihood based quantization loss, while an extra precision penalty term is introduced in the mixed precision NAS cost function. The larger performance difference between the mixed precision architecture search approach and the other two methods, for example, shown in Tables II and III, may be attributed to the use of Softmax function based architecture weights in the DARTS super-network. When similar architecture weights are obtained, the confusion over different candidate precision settings increases and search errors may occur.

Future research will focus on improving mixed precision quantization methods and their application to other neural network components of speech recognition systems and end-to-end based neural architectures.

References

[1] S. Katz, "Estimation of probabilities from sparse data for the language model component of a speech recognizer," *IEEE Trans. Acoust., Speech, Signal Process.*, vol. 35, no. 3, pp. 400–401, Mar. 1987.

[2] S. F. Chen and J. Goodman, "An empirical study of smoothing techniques for language modeling," *Comput. Speech Lang.*, vol. 13, no. 4, pp. 359–394, 1999.

[3] Y. Bengio *et al.*, "A neural probabilistic language model," *J. Mach. Learn. Res.*, vol. 3, pp. 1137–1155, 2003.

[4] H. Schwenk, "Continuous space language models," *Comput. Speech Lang.*, vol. 21, no. 3, pp. 492–518, 2007.

[5] T. Mikolov *et al.*, "Recurrent neural network based language model," in *Proc. Interspeech*, 2010, pp. 1045–1048.

[6] E. Arisoy *et al.*, "Deep neural network language models," in *Proc. NAACL-HLT 2012 Workshop*, 2012, pp. 20–28.

[7] H. Le, I. Oparin, A. Allauzen, J. Gauvain, and F. Yvon, "Structured output layer neural network language models for speech recognition," *IEEE Trans. Audio, Speech, Lang. Process.*, vol. 21, no. 1, pp. 197–206, Jan. 2013.

[8] M. Sundermeyer, H. Ney, and R. Schlüter, "From feedforward to recurrent LSTM neural networks for language modeling," *IEEE Trans. Audio, Speech, Lang. Process.*, vol. 23, no. 3, pp. 517–529, Mar. 2015.

[9] X. Chen, X. Liu, Y. Wang, M. J. F. Gales, and P. C. Woodland, "Efficient training and evaluation of recurrent neural network language models for automatic speech recognition," *IEEE Trans. Audio, Speech, Lang. Process.*, vol. 24, no. 11, pp. 2146–2157, Nov. 2016.

[10] X. Chen, X. Liu, Y. Wang, A. Ragni, J. H. M. Wong, and M. J. F. Gales, "Exploiting future word contexts in neural network language models for speech recognition," *IEEE Trans. Audio, Speech, Lang. Process.*, vol. 27, no. 9, pp. 1444–1454, Sep. 2019.

[11] K. Irie *et al.*, "Language modeling with deep transformers," in *Proc. Interspeech*, 2019, pp. 3905–3909.

[12] K. Li *et al.*, "An empirical study of transformer-based neural language model adaptation," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process.*, 2020, pp. 7934–7938.

[13] E. Beck *et al.*, "LVCSR with transformer language models," in *Proc. Interspeech*, 2020, pp. 1798–1802.

[14] P. Baquero-Arnal *et al.*, "Improved hybrid streaming ASR with transformer language models," in *Proc. Interspeech*, 2020, pp. 2127–2131.

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

[16] A. Vaswani *et al.*, "Attention is all you need," in *Proc. Neural Inf. Process. Syst.*, 2017, pp. 5998–6008.

[17] J. Xu, S. Hu, J. Yu, X. Liu, and H. Meng, "Mixed precision quantization of transformer language models for speech recognition," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process.*, 2021, pp. 7383–7387.

[18] J. Cheng *et al.*, "Long short-term memory-networks for machine reading," in *Proc. Conf. Empirical Methods Natural Lang. Process.*, 2016, pp. 551–561.

[19] Z. Lin *et al.*, "A structured self-attentive sentence embedding," in *Proc. Int. Conf. Learn. Representations*, 2017.

[20] A. Parikh *et al.*, "A decomposable attention model for natural language inference," in *Proc. Conf. Empirical Methods Natural Lang. Process.*, 2016, pp. 2249–2255.

[21] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2016, pp. 770–778.

[22] J. L. Ba *et al.*, "Layer normalization," 2016, *arXiv:1607.06450*.

[23] J. Gehring *et al.*, "Convolutional sequence to sequence learning," in *Proc. Int. Conf. Learn. Representations*, 2017, pp. 1243–1252.

[24] A. Zeyer, P. Bahar, K. Irie, R. Schlüter, and H. Ney, "A comparison of transformer and LSTM encoder decoder models for ASR," in *Proc. IEEE Autom. Speech Recognit. Understanding Workshop*, 2019, pp. 8–15.

[25] Y. Wang *et al.*, "Transformer-based acoustic modeling for hybrid speech recognition," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process.*, 2020, pp. 6874–6878.

[26] Q. Zhang *et al.*, "Transformer transducer: A streamable speech recognition model with transformer encoders and RNN-T loss," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process.*, 2020, pp. 7829–7833.

[27] S. Karita *et al.*, "A comparative study on transformer vs RNN in speech applications," in *Proc. IEEE Autom. Speech Recognit. Understanding Workshop*, 2019, pp. 449–456.

[28] A. Graves, "Sequence transduction with recurrent neural networks," in *Proc. Int. Conf. Mach. Learn.*, 2012.

[29] W. Chan, N. Jaitly, Q. Le, and O. Vinyals, "Listen, attend and spell: A neural network for large vocabulary conversational speech recognition," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process.*, 2016, pp. 4960–4964.

[30] Z. Tüske *et al.*, "Single headed attention based sequence-to-sequence model for state-of-the-art results on switchboard," in *Proc. Interspeech*, 2020, pp. 551–555.

[31] D. Soudry *et al.*, "Expectation backpropagation: Parameter-free training of multilayer neural networks with continuous or discrete weights," *Proc. Neural Inf. Process. Syst.*, vol. 1, pp. 963–971, 2014.

[32] M. Courbariaux *et al.*, "Binaryconnect: Training deep neural networks with binary weights during propagations," in *Proc. Neural Inf. Process. Syst.*, 2015, pp. 3123–3131.

[33] I. Hubara *et al.*, "Quantized neural networks: Training neural networks with low precision weights and activations," *J. Mach. Learn. Res.*, vol. 18, no. 1, pp. 6869–6898, 2017.

[34] M. Rastegari *et al.*, "Xnor-net: Imagenet classification using binary convolutional neural networks," in *Proc. Eur. Conf. Comput. Vis.*, Cham, Springer, 2016, pp. 525–542.

[35] Z. Dong *et al.*, "Hawq: Hessian aware quantization of neural networks with mixed-precision," in *Proc. IEEE/CVF Int. Conf. Comput. Vis.*, 2019, pp. 293–302.

[36] K. Wang, Z. Liu, Y. Lin, J. Lin, and S. Han, "HAQ: Hardware-aware automated quantization with mixed precision," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2019, pp. 8604–8612.

[37] Z. Dong *et al.*, "Hawq-v2: Hessian aware trace-weighted quantization of neural networks," in *Proc. Adv. Neural Inf. Process. Syst.*, 2019.

[38] S. Han *et al.*, "Learning both weights and connections for efficient neural networks," in *Proc. Neural Inf. Process. Syst.*, 2015, pp. 1135–1143.

[39] S. Han *et al.*, "Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding," in *Proc. Int. Conf. Learn. Representations*, 2015.

[40] H. Mao *et al.*, "Exploring the regularity of sparse structure in convolutional neural networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2017, pp. 1927–1934.

[41] Baoyuan Liu, Min Wang, H. Foroosh, M. Tappen, and M. Penksy, "Sparse convolutional neural networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2015, pp. 806–814.

[42] W. Wen *et al.*, "Learning structured sparsity in deep neural networks," *Neural Inf. Process. Syst.*, vol. 29, pp. 2074–2082, 2016.

[43] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the inception architecture for computer vision," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2016, pp. 2818–2826.

[44] C. Szegedy *et al.*, "Inception-v4, inception-resnet and the impact of residual connections on learning," *Proc. AAAI Conf. Artif. Intell.*, vol. 31, 2017, pp. 4278 4284.

[45] G. Hinton *et al.*, "Distilling the knowledge in a neural network," *Neural Inf. Process. Syst.*, 2014.

[46] A. Romero *et al.*, "Fitnets: Hints for thin deep nets," in *Proc. Int. Conf. Learn. Representations*, 2015.

[47] Y. Chebotar and A. Waters, "Distilling knowledge from ensembles of neural networks for speech recognition," in *Proc. Interspeech*, 2016, pp. 3439–3443.

[48] M. Jaderberg *et al.*, "Speeding up convolutional neural networks with low rank expansions," in *Proc. Brit. Mach. Vis. Conf.*, 2014.

[49] V. Lebedev *et al.*, "Speeding-up convolutional neural networks using fine-tuned CP-decomposition," in *Proc. Int. Conf. Learn. Representations*, 2015.

[50] C. Tai *et al.*, "Convolutional neural networks with low-rank regularization," in *Proc. Int. Conf. Learn. Representations*, 2016.

[51] V. Sindhwani *et al.*, "Structured transforms for small-footprint deep learning," in *Proc. Neural Inf. Process. Syst.*, 2015, pp. 3088–3096.

[52] F. N. Iandola *et al.*, "Squeezenet: Alexnet-level accuracy with 50x fewer parameters and< 0.5 mb model size," 2016, *arXiv:1602.07360*.

[53] X. Zhang, X. Zhou, M. Lin, and J. Sun, "Shufflenet: An extremely efficient convolutional neural network for mobile devices," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2018, pp. 6848–6856.

[54] N. Ma *et al.*, "ShuffleNet V2: Practical guidelines for efficient CNN architecture design," in *Proc. Eur. Conf. Comput. Vis.*, 2018, pp. 116–131.

[55] T. N. Sainath, B. Kingsbury, V. Sindhwani, E. Arisoy, and B. Ramabhadran, "Low-rank matrix factorization for deep neural network training with high-dimensional output targets," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process.*, 2013, pp. 6655–6659.

[56] Y. Wang *et al.*, "Small-footprint high-performance deep neural network-based speech recognition using split-VQ," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process.*, 2015, pp. 4984–4988.

[57] X. Liu *et al.*, "Binarized LSTM language model, "*Proc. Conf. North Amer. Chapter Assoc. Comput. Linguistics, Hum. Lang. Technol., vol. 1*, pp. 2113–2121, 2018.

[58] K. Yu, R. Ma, K. Shi, and Q. Liu, "Neural network language model compression with product quantization and soft binarization," *IEEE Trans. Audio, Speech, Lang. Process.*, vol. 28, pp. 2438–2449, Aug. 2020.

[59] Y. He *et al.*, "Streaming end-to-end speech recognition for mobile devices," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process.*, 2019, pp. 6381–6385.

[60] Ł. Dudziak *et al.*, "Shrinkml: End-to-end ASR model compression using reinforcement learning," in *Proc. Interspeech*, 2019, pp. 2235–2239.

[61] K. C. Sim *et al.*, "An investigation into on-device personalization of end-to-end automatic speech recognition models," in *Proc. Interspeech*, 2019, pp. 774–778.

[62] Y.-m. Qian and X. Xiang, "Binary neural networks for speech recognition," *Front. Inf. Technol. Electron. Eng.*, vol. 20, no. 5, pp. 701–715, 2019.

[63] K. O. Stanley and R. Miikkulainen, "Evolving neural networks through augmenting topologies," *Evol. Comput.*, vol. 10, no. 2, pp. 99–127, 2002.

[64] K. Kandasamy *et al.*, "Neural architecture search with Bayesian optimisation and optimal transport," in *Proc. Adv. Neural Inf. Process. Syst.*, 2018, pp. 2020–2029.

[65] B. Zoph and Q. V. Le, "Neural architecture search with reinforcement learning," in *Proc. Int. Conf. Learn. Representations*, 2017.

[66] B. Baker *et al.*, "Designing neural network architectures using reinforcement learning," in *Proc. Int. Conf. Learn. Representations*, 2017.

[67] H. Cai *et al.*, "Efficient architecture search by network transformation," *Proc. AAAI Conf. Artif. Intell.*, vol. 32, 2018, pp. 2787–2794.

[68] Z. Zhong, J. Yan, W. Wu, J. Shao, and C. Liu, "Practical block-wise neural network architecture generation," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2018, pp. 2423–2432.

[69] H. Liu *et al.*, "Darts: Differentiable architecture search," in *Proc. Int. Conf. Learn. Representations*, 2018.

[70] S. Xie *et al.*, "Snas: Stochastic neural architecture search," in *Proc. Int. Conf. Learn. Representations*, 2019.

[71] H. Cai *et al.*, "Proxylessnas: Direct neural architecture search on target task and hardware," in *Proc. Int. Conf. Learn. Representations*, 2019.

[72] Z. Cai and N. Vasconcelos, "Rethinking differentiable search for mixed-precision neural networks," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2020, pp. 2346–2355.

[73] S. Boyd *et al.*, *Distributed Optimization and Statistical Learning Via the Alternating Direction Method of Multipliers*. Now Publishers Inc, 2011.

[74] C. Leng *et al.*, "Extremely low bit neural network: Squeeze the last bit out with ADMM," *Proc. AAAI Conf. Artif. Intell.*, vol. 32, 2018, pp. 3466–3473.

[75] R. Ma, Q. Liu, and K. Yu, "Highly efficient neural network language model compression using soft binarization training," in *Proc. IEEE Autom. Speech Recognit. Understanding Workshop*, 2019, pp. 62–69.

[76] Y. Boo and W. Sung, "Fixed-point optimization of transformer neural network," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process.*, 2020, pp. 1753–1757.

[77] J. Xu, X. Chen, S. Hu, J. Yu, X. Liu, and H. Meng, "Low-bit quantization of recurrent neural network language models using alternating direction methods of multipliers," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process.*, 2020, pp. 7939–7943.

[78] D. Povey *et al.*, "Purely sequence-trained neural networks for ASR based on lattice-free MMI," in *Proc. Interspeech*, 2016, pp. 2751–2755.

[79] N. Dehak *et al.*, "Language recognition via i-vectors and dimensionality reduction," in *Proc. Interspeech*, 2011, pp. 857–860.

[80] P. Swietojanski and S. Renals, "Learning hidden unit contributions for unsupervised speaker adaptation of neural network acoustic models," in *Proc. IEEE Spoken Lang. Technol. Workshop*, 2014, pp. 171–176.

[81] J. J. Godfrey, E. C. Holliman, and J. McDaniel, "SWITCHBOARD: Telephone speech corpus for research and development," *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process.*, vol. 1, pp. 517–520, 1992.

[82] T. Hain *et al.*, "The AMI meeting transcription system: Progress and performance," in *Proc. Int. Workshop Mach. Learn. Multimodal Interact.*, Berlin, Heidelberg: Springer, 2006, pp. 419–431.

[83] S. Hochreiter and J. Schmidhuber, "LSTM can solve hard long time lag problems," in *Proc. Neural Inf. Process. Syst.*, 1997, pp. 473–479.

[84] D. Hendrycks and K. Gimpel, "Gaussian error linear units (gelus)," 2016, *arXiv:1606.08415*.

[85] F. Li *et al.*, "Ternary weight networks," 2016, *arXiv:1605.04711*.

[86] G. Korpelevi, "An extragradient method for finding saddle points and for other problems, konom. I mat," *Ekonom. i Mat. Metody*, no. 4, pp. 747–756, 1976.

[87] A. Stolcke, "Entropy-based pruning of backoff language models," in *Proc. DARPA Broadcast. News Transcription Understanding Workshop*, 1998.

[88] K. Seymore and R. Rosenfeld, "Scalable backoff language models," *Proc. 4th Int. Conf. Spoken Lang. Process.*, vol. 1, pp. 232–235, 1996.

[89] H. Avron and S. Toledo, "Randomized algorithms for estimating the trace of an implicit symmetric positive semi-definite matrix," *J. ACM*, vol. 58, no. 2, pp. 1–34, 2011.

[90] T. Elsken *et al.*, "Neural architecture search: A survey," *J. Mach. Learn. Res.*, vol. 20, no. 55, pp. 1–21, 2019.

[91] S. Hu *et al.*, "Neural architecture search for LF-MMI trained time delay neural networks," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process.*, 2020, pp. 6758–6762.

[92] S. Hu *et al.*, "Dsnas: Direct neural architecture search without parameter retraining," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2020, pp. 12081–12089.

[93] H. Pham *et al.*, "Efficient neural architecture search via parameters sharing," in *Proc. Int. Conf. Mach. Learn.*, 2018, pp. 4095–4104.

[94] A. Fasoli *et al.*, "4-Bit quantization of LSTM-Based speech recognition models," in *Proc. Interspeech*, 2021, pp. 2586–2590.

[95] D. Povey *et al.*, "The Kaldi speech recognition toolkit," in *Proc. IEEE Workshop Autom. Speech Recognit. Understanding*, 2011.

[96] X. Xie, X. Liu, T. Lee, and L. Wang, "Bayesian learning for deep neural network adaptation," *IEEE/ACM Trans. Audio, Speech, Lang. Process.*, vol. 29, pp. 2096–2110, May 2021.

[97] J. Li *et al.*, "On the comparison of popular end-to-end models for large scale speech recognition," in *Proc. Interspeech*, 2020, pp. 1–5.

[98] C. Liu *et al.*, "Progressive neural architecture search," in *Proc. Eur. Conf. Comput. Vis.*, 2018, pp. 19–34.

[99] A. M. Maddison, J. Chris, and Y. W. Teh, "The concrete distribution: A continuous relaxation of discrete random variables," in *Proc. Int. Conf. Learn. Representations*, 2017.

Jianwei Yu received the B.Sc. degree in physics from Nanjing University, Nanjing, China, in 2017. He is currently working toward the Ph.D. degree with the Chinese University of Hong Kong, Hong Kong. His current research interests include noise robust speech recognition, audio-visual speech recognition, Bayesian modeling, and language model.

Shoukang Hu (Member, IEEE) received the B.E. degree in mechanical and electrical engineering from the University of Electronic Science and Technology of China, Chengdu, China, in 2017. He is currently working toward the Ph.D. degree with the Chinese University of Hong Kong, Hong Kong. His current research interests include speech recognition, Bayesian modeling, and neural architecture search.
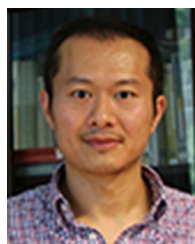
Xunying Liu (Member, IEEE) received the Ph.D. degree in speech recognition and the M.Phil. degree in computer speech and language processing from the University of Cambridge, Cambridge, U.K. He was a Senior Research Associate with Machine Intelligence Laboratory, Cambridge University Engineering Department and since 2016, he has been an Associate Professor with the Department of Systems Engineering and Engineering Management, Chinese University of Hong Kong, Hong Kong. His current research interests include machine learning, large vocabulary continuous speech recognition, statistical language modelling, noise robust speech recognition, audio-visual speech recognition, computer aided language learning, speech synthesis, and assistive technology. He and his students was the recipient of a number of best paper awards and nominations, including the Best Paper Award at ISCA Interspeech 2010 for the paper titled Language Model Cross Adaptation for LVCSR System Combination, and the Best Paper Award at IEEE ICASSP2019 for their paper titled BLHUC: Bayesian Learning of Hidden Unit Contributions for Deep Neural Network Speaker Adaptation. Dr. Xunying Liu is a Member of ISCA.

Helen Meng (Fellow, IEEE) received the B.S., M.S., and Ph.D. degrees in electrical engineering from the Massachusetts Institute of Technology, Cambridge, MA, USA. In 1998, she joined the Chinese University of Hong Kong, Hong Kong, where she is currently the Chair Professor with the Department of Systems Engineering and Engineering Management. She was the former Department Chairman and an Associate Dean of Research with the Faculty of Engineering. Her research interests include human–computer interaction via multimodal and multilingual spoken language systems, spoken dialog systems, computer-aided pronunciation training, speech processing in assistive technologies, health-related applications, and Big Data decision analytics. Between 2009 and 2011, she was the Editor-in-Chief of the IEEE TRANSACTIONS ON AUDIO, SPEECH AND LANGUAGE PROCESSING. She was the recipient of the IEEE Signal Processing Society Leo L. Beranek Meritorious Service Award in 2019. She was also on the Elected Board Member of the International Speech Communication Association (ISCA) and an International Advisory Board Member.

Junhao Xu received the B.Sc. degree in computer science from Shanghai Jiaotong University, Shanghai, China, in 2018. She is currently working toward the Ph.D. degree with the Chinese University of Hong Kong, Hong Kong. Her current research interests include model compression, audio-visual speech recognition, and language model.