Extremely Low Footprint End-to-End ASR System for Smart Device

Zhifu Gao1*, Yiwu Yao2*, Shiliang Zhang1, Jun Yang3, Ming Lei1, Ian McLoughlin3

1Speech Lab, Alibaba Group, China
2Platform of A. I., Alibaba Group, China
3ICT Cluster, Singapore Institute of Technology, Singapore

{zhifu.gzf, yiwu.yyw, sly.zsl, muzhuo.yj, lm86501}@alibaba-inc.com, ian.mcloughlin@singaporetech.edu.sg

Abstract

Recently, end-to-end (E2E) speech recognition has become popular, since it can integrate the acoustic, pronunciation and language models into a single neural network, which outperforms conventional models. Among E2E approaches, attention-based models, e.g. Transformer, have emerged as being superior. Such models have opened the door to deployment of ASR on smart devices, however they still suffer from requiring a large number of model parameters. We propose an extremely low footprint E2E ASR system for smart devices, to achieve the goal of satisfying resource constraints without sacrificing recognition accuracy. We design cross-layer weight sharing to improve parameter efficiency and further exploit model compression methods including sparsification and quantization, to reduce memory storage and boost decoding efficiency. We evaluate our approaches on the public AISHELL-1 and AISHELL-2 benchmarks. On the AISHELL-2 task, the proposed method achieves more than 10× compression (model size reduces from 248 to 24MB), at the cost of only minor performance loss (CER reduces from 6.49% to 6.92%).

Index Terms: speech recognition, san-m, weight sharing, sparsification, quantization

1. Introduction

There is growing interest in building automatic speech recognition (ASR) systems on smart devices to satisfy privacy, security and network bandwidth constraints. Recent advances in end-to-end ASR (E2E-ASR) means such systems are now strong candidates for such deployments. Compared to conventional hybrid ASR systems, E2E-ASR systems fold the acoustic, language and pronunciation models into a single sequence-to-sequence (seq2seq) model. Currently, there exist three popular E2E approaches, namely connectionist temporal classification (CTC) [1], recurrent neural network transducers (RNN-T) [2], and attention based encoder-decoders (AED) [3–5]. Unlike CTC-based models, RNN-T and AED have no independence assumption, and can achieve state-of-the-art performance without an external language model, making them more suitable for on-device deployment.

Typical AED models such as LAS [5] and Transformer [6], consist of an encoder and a decoder. The encoder transforms raw acoustic features into a high-level representation, while the decoder predicts output symbols in an auto-regressive manner. Transformer-based models have dominated seq2seq modeling in the ASR field, due to their superior recognition accuracy [7–14]. However, the large number of model parameters these ASR models require has become the main challenge for deployment in resource constrained scenarios, where both memory and computation resources are limited. Recently, researchers have made more effort to optimize ASR systems for smart device. For example, knowledge distillation [15], low-rank factorization [16], model sparsification [17] and network architecture search [18], etc. Such efforts have made some progress, nevertheless widespread deployment of ASR on smart devices remains a challenge; both in terms of memory and computational resource constraints.

In this work, we propose an extremely low footprint E2E ASR system for smart devices, to better trade-off computation resources and recognition accuracy. Firstly, motivated by ALBERT in neural language processing [19], we adopt cross-layer weight sharing to improve parameter efficiency. This prevents the number of model parameters from growing with the depth of the network, without seriously impacting performance. In addition to weight sharing for the ASR models, we incorporate model sparsification [20–24] and post-training quantization (PTQ) [25–30] methods for extreme model compression.

As shown in Fig. 1, the architecture of the proposed weight-shared ASR model contains an encoder and decoder, which adopts memory equipped self-attention (SAN-M) [31] and a feed-forward network (FFN) as the basic sub-layer. SAN-M and FFN are designed to be shared across different layers, to improve the parameter efficiency. The ASR model is pre-trained before fine-tuning with model sparsification. Then during sparse training, the weight importance, with Taylor expan-
In short, the storage size of the ASR model can be significantly reduced by weight sharing across sub-layers within each building block. As noted previously, further compression is achieved by two automatic model compression methods, namely sparsification and quantization. These two model-independent compression methods are orthogonally applied.

2.2. Model Sparsification

According to different sparse patterns, model sparsification methods can be classified into structural [20] or non-structural methods [21–24]. Non-structural sparsification randomly zeros parameters at the fine-grained level of the weights, leading to favourable accuracy robustness with high compression ratio. Therefore, in order to maintain the recognition accuracy of ASR models with high sparsity, non-structural sparsification is adopted in this work. The Taylor method [20], leveraging the second order expansion of the pruning cost as the importance metric, is then used to reveal the weight importance:

$$I = (w \cdot g)^2 = \left( w \cdot \frac{\partial L}{\partial w} \right)^2$$

In Eq. 1, the backward gradient information $g$ is applied for the implication of weight importance, thus the redundancy of model parameters can be effectively explored for sparsification. To enhance the effectiveness of sparse training with the Taylor metric, four aspects are considered:

- First, the weight importance is initialized as the weight magnitude;
- During training, the weight importance is updated with the Taylor method and EMA (exponential moving average);
- The compression ratio for each layer is dynamically determined with the overall ranking of weight importance;
- The compression ratio for the entire model is then progressively increased during sparse training.

Thereby the weight importance is updated at the training step $t$ using the following expression ($\alpha = 0.99$):

$$I_t = \begin{cases} \|w_t\|, & t = 0 \\ \alpha \cdot I_{t-1} + (1 - \alpha) \cdot (w_t \cdot g_t)^2, & t > 0 \end{cases}$$

2.3. Post-training Quantization

After sparse training of the weight-shared ASR model, the quantization is applied for further compression and acceleration. Post-training quantization (PTQ) is preferred in this work for a simpler compression flow. Meanwhile, symmetric quantization is exploited to map the activations and weights into an 8-bit integer range $[-127, 127]$. The quantization function $Q(v)$ for a given tensor $v$ is expressed as follows:

$$Q(v) = \text{round}(\text{clip}(v/s, -T_s, T_s))$$

Where $s$ is the scaling factor for quantization, and $T_s = 127$ denotes the integer range. According to Eq. 3, quantization noise is inevitably introduced to the quantized ASR model, thanks to the influence of clipping and rounding errors.

In order to minimize the quantization noise and thus maintain recognition accuracy, several PTQ strategies are explored in this work, including a refined KL algorithm, improved ADMM and label-free automatic mixed precision (AMP) quantization.
Fig. 3: Two histogram types for the KL algorithm: (a) absolute values are mapped into $[0, 127]$; (b) the real values are mapped into $[0, 255]$. The scaling factor is calculated as $T/127$ of the statistical threshold $T$ in the KL algorithm.

2.3.1. Refined KL Algorithm

The KL algorithm is used to calculate the scaling factors for the activation quantization process, as detailed in [27]. As shown in Fig. 3a, the absolute value of activations (excluding zeros) are quantized and then mapped into a histogram with an 8-bit integer range $[0, 127]$, that is used for the quantization of ReLU activations.

Adhering to the basic principle of the KL algorithm, we refine the process by collecting the histogram over the original real activation values (excluding zeros). As shown in Fig. 3b, the integer range is now $[0, 255]$, which broadens the histogram and captures finer statistics, meaning that the quantization resolution is nearly doubled. This improvement is beneficial in reducing quantization noise associated with activations, especially in cases where activations are transferred without ReLU, which commonly occurs in the attention modules of Transformer-based ASR models.

2.3.2. Improved ADMM

Furthermore, to reduce noise due to weight quantization, ADMM [28] is utilized to optimize scaling factors applied prior to the quantization, according to the following iterative formula:

$$s_{k+1} = \frac{w \cdot E(w/s_k)}{E(w/s_k) - E(w/s_k)}$$

(4)

Where $k$ denotes the iteration step, $w$ represents a weight tensor, and $E(\cdot)$ is the expectation. In order to obtain the best scaling factor as the local optimum, we improve ADMM by adopting a winner-take-all method. The mean squared error (MSE) of the original weight and quantized weight are recorded for each iteration. Finally, the scaling factor which results in the minimum MSE, is chosen for weight quantization as follows:

$$s_i = \arg \min_i \|w - s_i \cdot Q(w)\|^2$$

(5)

2.3.3. Label-free Automatic Mixed Precision

After post-training quantization with the refined KL algorithm and ADMM, the quantization noise for activations and weights can be eliminated to a certain extent. However, we have found that accuracy loss may still exist due to occasional abnormal network layers within the ASR model. To overcome this issue, we apply label-free automatic mixed precision (AMP) quantization as a fallback. This overcomes the relatively higher quantization cost for those few abnormal layers.

As demonstrated in Fig. 4, we build a sub-graph of fake quantization and MSE calculation in parallel to each network layer (e.g. Conv1D), to collect corresponding layer-wise MSE as the quantization cost. The calibration-set for KL algorithm, and scaling factors for quantization, are reused here for the statistics of MSE. Then the layer-wise MSE is calculated as:

$$\text{Cost}_j = \|H_j(a, w) - DQ(H_j(Q(a), Q(w))))\|^2$$

(6)

Where $H_j$ denotes the $j$-th layer operation (e.g. Conv1D), $DQ(\cdot)$ means the de-quantization function, and $a, w$ are the activations and weights respectively. Abnormal layers with Top-K quantization costs can, as fallback, use a float32 implementation. Since there is no need for data labels to compute the accuracy difference on an evaluation-set for the decision of mixed-precision setting, proposed AMP method is label-free. Additionally, compared with alternative label-free AMP methods [30] requiring a greedy search for mixed-precision settings, the time cost of our layer-wise AMP method is much lower.

3. Experiments

3.1. Experimental Setup

We have evaluated the proposed method on two Mandarin speech recognition tasks, the 170-hour AISHELL-1 [33] and the 1000-hour AISHELL-2 task [34]. For AISHELL-2, we use all the training data (1000 hours) for training, dev\_jos and test\_jos sets for validation and evaluation, respectively. Acoustic features used in all experiments are 80-dimensional log-mel filter-bank (FBK) energies computed on 25ms windows with 10ms shift. We stack the consecutive frames within a context window of $7 \cdot 3 + 1 + 3$ to produce the 560-dimensional features and then down-sample the input frame rate to 60ms. Acoustic modeling units are Chinese characters, totalling the vocabulary size of 4233 both for AISHELL-1 and AISHELL-2. Models are trained with Tensorflow [35] and 8 Nvidia GPUs. As to the detailed experimental setup, we adopt LazyAdamOptimizer with $\beta_1 = 0.9, \beta_2 = 0.999$, and the strategy for learning rate is noam\_decay\_v2 with $d_{model} = 512$, $warmup\_st = 8000, k = 4$. Label smoothing and dropout regularization with a value of 0.1 are incorporated to prevent over-fitting. SpecAugment [36] is also used for data augmentation in all experiments.

3.2. On AISHELL-1

In this subsection, we will evaluate the performance of weight sharing on the AISHELL-1 task, detailed in Table 1. Symbols $M$ and $N$ denote that the model contains $M$ encoder blocks and $N$ decoder blocks, as shown in Fig. 1. $S_1/S_2$ refers to the number of weight-shared basic sub-layers.
The top half of Table 1 (EXP0 to 4) are baselines with different numbers of encoder blocks, each containing only one sub-layer, without weight sharing. As expected, performance increases with encoder depth (from 2 to 9), but model size also increases (from 76 to 160MB). The bottom half of the table (EXP7 to 9) shows the effect when several sub-layers within each block share the same weights. Performance again improves with encoder depth (from 6 to 12), but model size is constrained to 88MB. Comparing EXP4 with EXP8, both of the same depth, weight sharing reduces the EXP4 model size from 160MB to 88MB at the cost of only small performance loss (CER from 6.49% to 6.92%). In addition, compared to only using weight sharing in the encoder, EXP5 and EXP6 results reveal that weight sharing applied to the decoder can achieve a smaller but better model.

From the above experiments, we see that weight sharing prevents the model parameters from growing with the depth of the network, without seriously hurting performance.

3.3. On AISHELL-2

We now evaluate the performance of weight sharing on the AISHELL-2 task, detailed in Table 2. \( N, M \) and \( S_1/S_2 \) have the same meaning as in Sec. 3.2.

The EXP0 baseline model has 40 encoder sub-layers and 12 decoder sub-layers. Compared to EXP10, the proposed model in EXP11 obtains comparable performance with 1.6× size reduction (from 248 to 151MB), thanks to cross-layer weight sharing. Combining weight sharing with sparsification and quantization, more than 10× compression (model size from 248 to 24MB) is achieved, with only small performance loss (CER from 6.49% to 6.92%).

We further evaluate sparsification effectiveness. Using the pre-trained weight-shared ASR model (EXP11 in Table 2), we conduct sparse training with a sparsity of 30%, 40% and 50%. Table 3 shows that the storage size of the ASR model effectively reduces with increasing sparsity. At 50% sparsity, we obtain 2.1× compression (model size from 151 to 72MB) with a small accuracy loss of 0.41% (CER from 6.54% to 6.95%).

Lastly, we evaluate a PTQ strategy. After sparsification, we introduce the selected combination of PTQ strategies to quantize the ASR model with little performance decay. As illustrated in Tab. 4, the KL algorithm is used as the basic PTQ method, while ADMM can be adopted to minimize weight quantization noise. As a result, we can obtain an extremely compressed ASR model with a size of just 24MB and a CER of 6.92% on AISHELL-2.

To further optimize the quantization, we evaluate the label-free AMP method of Section 2.3.3 on the 50% sparsity ASR model. As listed in Table 4, through the fallback of three network layers with higher MSE defined in Eq. 6, the performance of the quantized model is improved. Meanwhile, we can see that AMP based on the joint usage of KL and ADMM achieves an improvement, due to the better Pareto efficiency in the trade-off between model size and accuracy.

4. Conclusions

This work proposed an extremely low footprint E2E ASR system for smart devices, to achieve the goal of satisfying resource constraints without sacrificing recognition accuracy. We adopt cross-layer weight sharing to improve parameter efficiency. This technique prevents the model parameters from growing with the depth of the network, which reduces the number of parameters without seriously affecting performance. We further utilize model compression methods including sparsification and quantization, to reduce memory storage requirements and boost decoding efficiency on smart devices. We evaluated our approach on public AISHELL-1 and AISHELL-2 benchmarks. On the AISHELL-2 task, the proposed method achieves more than 10× compression (model size from 248 to 24MB) with small accuracy loss (CER from 6.49% to 6.92%).

In future work, we intend to further explore more effective model compression methods such as structural sparsification and joint optimization of PTQ strategies, and evaluate the methods on more scenarios.
5. References


