INTERSPEECH 2020
October 25–29, 2020, Shanghai, China

Quantization Aware Training with Absolute-Cosine Regularization for Automatic Speech Recognition

Hieu Duy Nguyen, Anastasios Alexandridis, and Athanasios Mouchtaris

Alexa Machine Learning, Amazon.com, USA
{hieng, aanast, mouchta}@amazon.com

Abstract

Compression and quantization is important to neural networks in general and Automatic Speech Recognition (ASR) systems in particular, especially when they operate in real-time on resource-constrained devices. By using fewer number of bits for the model weights, the model size becomes much smaller while inference time is reduced significantly, with the cost of degraded performance. Such degradation can be potentially addressed by the so-called quantization-aware training (QAT). Existing QATs mostly take into account the quantization in forward propagation, while ignoring the quantization loss in gradient calculation during back-propagation. In this work, we introduce a novel QAT scheme based on absolute-cosine regularization (ACosR), which enforces a prior, quantization-friendly distribution to the model weights. We apply this novel approach into ASR task assuming a recurrent neural network transducer (RNN-T) architecture. The results show that there is zero to little degradation between floating-point, 8-bit, and 6-bit ACosR models. Weight distributions further confirm that in-training weights are very close to quantization levels when ACosR is applied.

Index Terms: speech recognition, quantization-aware training (QAT), recurrent neural network transducer (RNN-T), regularization, absolute-cosine regularization

1. Introduction

For various applications, neural network (NN) models are often required to be executed fast so that user perceived latency is small. This is especially important for applications with back-and-forth interactions with customers, such as virtual assistants. However, during training time, the NN model weights/variables are mostly represented in 32-bit/floating-point format. The resulting model typically has a very large number of parameters and is slow during inference time. It is thus important that the NN is compressed and quantized to reduce the model footprint. For example, post-training quantization is often applied to NN models, i.e. each weight/variable is quantized from 32 down to 16, 8, or even 4 bits. By operating with a smaller number of bits, the model becomes smaller and faster, thus consumes less memory and is able to facilitate real-time user interactions.

Unsurprisingly, post-training compression and quantization usually leads to, sometimes significant, performance degradation for NNs. One of the solutions to this problem is to employ the so-called quantization-aware training (QAT) schemes. The intuition is that during model training, the effect of compression/quantization will be taken into account, thus mitigating the accuracy loss introduced by post-training processing.

Due to its importance, QAT has attracted significant attention and various schemes have been proposed. In [1, 2, 3], the authors introduced a fake-quantizer and a hashing function to mimic the quantization effect during model training. Another approach is to leverage on Gumbel-softmax trick, which is capable of learning discrete model weights [4]. In [5], Georges *et al.* proposed to use Gaussian mixture to train and obtain networks with binary weight values. Most of the existing work has focused on replicating the compression/quantization effect in the forward propagation, under which the model loss is evaluated. It is assumed that such loss already incorporates the quantization effect, leading to no quantization-related term in the back propagation step. As a consequence, quantization is only loosely coupled into the gradient evaluation and weight update. Another disadvantage of existing QAT schemes is high memory consumption and intensive computation. This work presents a novel QAT approach that efficiently addresses such problems.

Until recently, a common model architecture for ASR is the deep neural network (DNN) - Hidden Markov Model (HMM) approach, in which the acoustic model predicts the senones/phonemes of each audio frame, while the language model decides the sentence hypothesis [6, 7]. DNN-HMM requires multiple intermediate components, such as lexicon model and decision tree, that in turn depend on domain-knowledge and human transcription effort. End-to-end (E2E) systems have significantly reduced much of the dependencies, by directly transducing the acoustic feature input into character/word/word-piece sequence output. Some of the most important approaches are Connectionist Temporal Classification (CTC) [8, 9], Listen-Attend-and-Spell (LAS) [10], and recurrent neural network transducer (RNN-T) [9, 11]. Among those emerging architecture, RNN-T has attracted significant attention, due to its online processing capability (in contrast with LAS) and no requirement for audio-output mapping (in contrast with CTC). As a consequence, we thus leverage the RNN-T for our QAT experiments. Note that our QAT approach is applicable to other NN models with little modification.

In this work, we present a novel approach for QAT via regularization. The intuition is to impose on the model weights a distribution which has a similar shape to that of a quantized model. We achieve this by devising a new function called absolute-cosine regularization (ACosR). The total loss thus includes both accuracy and ACosR losses, in which the ACosR loss measures the similarity between our model and a quantized one. We demonstrate the effectiveness of our approach by applying it into an ASR system with RNN-T architecture. In-training, the weight distribution shows that ACosR reliably forces the weights to converge to the quantization levels, while introducing zero to very little performance degradation for 8-bit QAT. Investigation for 6-bit QAT models also shows a slight degradation compared to floating-point baseline.

2. Related Works

Studies have shown that the NN footprint can be reduced significantly with moderate performance degradation for various tasks such as image recognition [3, 12], natural language un-

http://dx.doi.org/10.21437/Interspeech.2020-1991

derstanding [4] or speech recognition [13]. Various techniques have been proposed, including integer quantization [2], hashing [3], vector quantization [14] and matrix factorization [15, 16].

To further improve the performance, various QAT approaches have been proposed over the years. One research direction is to simulate the compression and quantization effect during training, by incorporating integer quantization into the NN structure [1, 2, 17]. Such approach has often been mentioned as "fake quantization" [18]. The main idea is to calculate the loss assuming a quantized NN under forward propagation. Since it is difficult to compute the weights' gradient under quantization, most studies usually consider straight-through estimators (STE) by ignoring quantization-related gradient term in back propagation [18]. Under such assumption, the approach is not strictly QAT anymore. Consider a toy example of $\min_{w\in\{0,1\}} f_{\text{te}}(w)$ in which

$$f_{\text{te}}(w) = \begin{cases} 3(w-0.4)^2, & w \leq 0.4, \\ (w-0.4)^2, & w > 0.4. \end{cases} \tag{1}$$

It is observed that the solution of $\min_{w\in\{0,1\}} f_{\text{te}}(w)$ is $w^* = 1$. Under fake quantization, since the quantization loss is not included into gradient calculation, the problem becomes $w^*_{fq} = \text{quantize}_{w\in\{0,1\}}\left[\arg\min_w f_{\text{te}}(w)\right] = 0$, which obviously is an incorrect solution.

The compression and quantization of NNs can also be achieved by viewing it as finding a codebook and corresponding code for each weight value [4, 19, 20]. Notably in [4], Shu *et al.* introduced the Gumbel-softmax trick to compress a word embedding for the natural language understanding task, thus enabling the optimization of discrete weight values. Such approaches, however, impose significant memory consumption and computation since we need to learn both the codebook and compositional codes of each weight.

The idea of applying regularization to improve post-training quantization has been explored in [21, 22], in which the authors applied multiple functions such as L2 and sawtooth, and observed which function returns the best model after post-training quantization. In a recent work [5], Georges *et al.* introduced a Gaussian mixture regularization with two Gaussian functions to learn a binarized NN, which achieves little performance degradation versus floating-point model for an intent classification task. Note that using a Gaussian mixture will require intensive gradient computation, since each Gaussian function will need to be evaluated for each model variable. For example, an ASR system with 10-million parameters and 8-bit quantization will require 2^8 Gaussian functions and $2^8 \times 10 \times 10^6 = 2.56$ billion weight gradient calculations. Using Gaussian mixtures for QAT is thus not suitable for many deep learning tasks.

3. Absolute-Cosine Regularization

The two main limitations of existing QAT approaches, namely, no quantization-related gradient term and intensive computation/memory requirement, can be addressed efficiently with ACosR. In this section, we describe the intuition and implementation of ACosR. Given a set of model weights $\mathcal{W} = \{w_1, w_2, ..., w_N\}$, a compression/quantization scheme will convert $\mathcal{W}$ into $\mathcal{W}_q = \{q(w_1), q(w_2), ..., q(w_N)\}$. Here $q(x)$ is a function that quantizes/compresses a value x into one of the quantization levels $\mathcal{Q} = \{q_1, q_2, ..., q_K\}$. A quantized weight distribution thus takes the form of $\Psi_{\text{quantized}}(x) = \sum_{k=1}^{K} \alpha_k \delta(x, q_k)$, in which $\{\alpha_k\}_{k=1}^{K}$ are the scaling factors

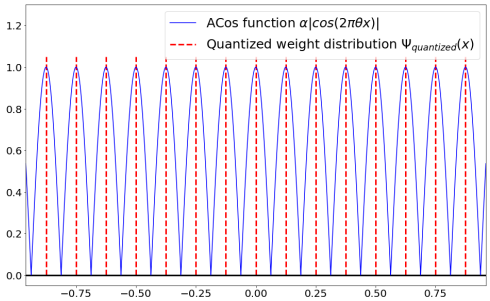

Figure 1: *Comparison between* $\Psi_{quantized}(x)$ *with* $\alpha_k = 1, \forall k$, $\mathcal{Q}_{linear} = \{..., -\frac{2}{8}, -\frac{1}{8}, 0, \frac{1}{8}, \frac{2}{8}, ...\}$, *and* $ACos(x)$ *with* $\alpha = 1, \theta = 8$. *The shape of* $e^{ACos(x)}$ *is similar to that of* $ACos(x)$.

and the delta function is defined as

$$\delta(x, q) = \begin{cases} 1, & x = q, \\ 0, & otherwise. \end{cases} \tag{2}$$

To achieve fast inference and low user-perceived latency (UPL), most of NN models employ linear quantization schemes. In that case, $\mathcal{Q}_{\text{linear}} = \{..., -2q, -q, 0, q, 2q, ...\}$. Fig. 1 shows an example of a quantized weight distribution $\Psi_{\text{quantized}}(x)$ with $\alpha_k = 1, \forall k$ when $\mathcal{Q}_{\text{linear}} = \{..., -\frac{2}{8}, -\frac{1}{8}, 0, \frac{1}{8}, \frac{2}{8}, ...\}$. In ACosR approach, we seek to enforce such quantized weight distribution upon the weights during training. Recall that,

$$\begin{aligned} \mathbf{w}^{\text{best}} &= \arg\max_{\mathbf{w}} \mathcal{P}(\mathbf{w}|\text{observed data}) \\ &= \arg\max_{\mathbf{w}} \frac{\mathcal{P}(\text{observed data}|\mathbf{w})\mathcal{P}(\mathbf{w})}{\mathcal{P}(\text{observed data})} \\ &= \arg\max_{\mathbf{w}} \left[\ln \mathcal{P}(\text{observed data}|\mathbf{w}) + \ln \mathcal{P}(\mathbf{w})\right] \\ &= \arg\min_{\mathbf{w}} \left[\mathcal{L}_{\text{accuracy}}(\mathbf{w}) + \mathcal{L}_{\text{regularization}}(\mathbf{w})\right]. \end{aligned} \tag{3}$$

Here, $\mathbf{w}^{\text{best}}$, $\mathcal{P}(.)$, $\mathcal{P}(.|.)$, $\mathcal{L}_{\text{accuracy}}(\mathbf{w})$, $\mathcal{L}_{\text{regularization}}(\mathbf{w})$, and $\ln$ denote the best weights, probability of an event, conditional probability of an event given another, model loss due to accuracy, model loss due to regularization, and natural logarithm, respectively. The weight update at each training step is

$$\mathbf{w} \leftarrow \mathbf{w} - \text{learning-rate} \times \left\{ \frac{\partial \mathcal{L}_{\text{accuracy}}(\mathbf{w})}{\partial \mathbf{w}}\bigg|_{\mathbf{w}} + \frac{\partial \mathcal{L}_{\text{regularization}}(\mathbf{w})}{\partial \mathbf{w}}\bigg|_{\mathbf{w}} \right\}. \tag{4}$$

Eq. (3) thus suggests that we can compel a weight distribution by introducing an appropriate regularization function. The main difficulty is to select a function $f(x)$ in which $\exp(f(x))$ has similar shape as that of $\Psi_{\text{quantized}}(x)$, and $f(x)$ requires minimal additional computation as well as memory consumption.

For QAT, we achieve such goal by using ACosR

$$\mathcal{L}_{\text{regularization}}(\mathbf{w}) = \mathcal{L}_{\text{acosr-reg}}(\mathbf{w}) = -\text{ACos}(\mathbf{w}), \tag{5}$$

in which the absolute-cosine function is defined as

$$\text{ACos}(x) = \alpha \left|\cos(\pi\theta x)\right|, \tag{6}$$

with α being a scaling factor and θ being the cosine frequency, to be defined based on a specific linear quantization scheme. Fig. 1 visualizes $\text{ACos}(x)$ and compares it with the desirable, quantized weight distribution $\Psi_{\text{quantized}}(x)$. Note that the shape of $e^{ACos(x)}$ after being scaled is similar to that of $\text{ACos}(x)$.

Another perspective for using $\text{ACos}(x)$ is that it approximates the quantization loss. In particular, it is interesting to compare $\mathcal{L}_{\text{acosr-reg}}(x) = -\text{ACos}(x)$ with the L2 loss $\mathcal{L}_{\text{L2-quant}}(x) = \alpha_{\text{L2}} \left|x - q(x)\right|^2$, which measures how close a

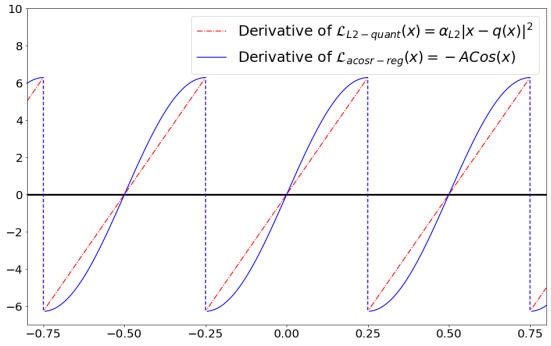

Figure 2: *Derivative of* $\mathcal{L}_{acosr\text{-}reg}(x)$ *and* $\mathcal{L}_{L2\text{-}quant}(x)$ *with* $\theta = 2$, $\alpha = 1$ *and* $\alpha_{L2} = \pi$.

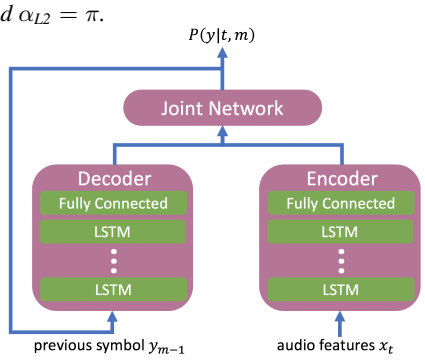

Figure 3: *Model diagram of RNN-T. Here the joint-network might have different architectures.*

weight x is to its quantized value $q(x)$. Note that it is difficult to implement $\mathcal{L}_{\text{L2-quant}}$, as it requires a mixture of L2 regularization centering at quantization levels. Similar to Gaussian mixture approach [5], that leads to intensive gradient computation and memory consumption.

We first observe that as a weight value approaches one of the quantization level q_k, both function $\mathcal{L}_{\text{acosr-reg}}(x)$ and $\mathcal{L}_{\text{L2-quant}}(x)$ becomes smaller and smaller. We further compare the derivatives of $\mathcal{L}_{\text{acosr-reg}}(x)$ and $\mathcal{L}_{\text{L2-quant}}(x)$, which are derived and illustrated in Eq. (7) and Fig. 2:

$$\frac{\partial \mathcal{L}_{\text{acosr-reg}}(x)}{\partial x} = \begin{cases} \alpha\pi\theta \sin(\pi\theta x), & x \in \left[\frac{4m-1}{2\theta}, \frac{4m+1}{2\theta}\right], \\ & m \in \{..., -1, 0, 1, 2, ...\} \\ -\alpha\pi\theta \sin(\pi\theta x), & otherwise. \end{cases}$$

$$\frac{\partial \mathcal{L}_{\text{L2-quant}}(x)}{\partial x} = 2\alpha_{\text{L2}}\left(x - q(x)\right) \tag{7}$$

As shown in Fig. 2, the derivative is positive when a weight value is near but larger than a quantization level, and negative in the opposite case. From Eq. (4), it is apparent that the weight values will be driven towards the quantization levels by using ACosR. Furthermore, we notice a significant similarity between the derivatives of $\mathcal{L}_{\text{acosr-reg}}(x)$ and $\mathcal{L}_{\text{L2-quant}}(x)$. This confirms that ACosR is an effective measurement of how quantization-friendly a model is.

Finally, it is observed that $\text{ACos}(x)$ is almost everywhere differentiable and quasi-convex, thus guaranteeing the weight convergence. Furthermore, since we only leverage on one function for regularization, the gradient computation and weight update will require small additional memory consumption and computing resource. This fact, as discussed above, is very important to machine learning applications that have large numbers of weights such as ASR or Machine Translation.

Table 1: *Joint network (JN) architectures and numbers of word-pieces (WPs) under different investigated scenarios.*

	Data	JN	# of WPs
Setup 1	English, Librispeech	one FC layer	2.5K
Setup 2	English	a simple addition operator	10K
Setup 3	French	a simple addition operator	10K

Table 2: *Amount of training and testing data (hours)*

	Training	Testing			
		Freq.		Apps.	Enterm.
English	23K	115		30	26
French	10K	98		41	38
		dev-clean	dev-other	test-clean	test-other
Librispeech	960	5.4	5.3	5.4	5.1

4. Experiment Setup

4.1. Model architecture

In this work, we apply QAT-ACosR into ASR task and study its effectiveness. In particular, we consider an ASR system based on the RNN-T architecture shown in Fig. 3. An RNN-T model consists of 4 main components: encoder, decoder, joint network, and a word-piece model which provides the words for decoding hypotheses. The encoder extracts acoustic features from the audio input x_t, while the decoder learns the relation between different output symbols. Loosely speaking, the encoder and decoder play the roles of acoustic and language model in conventional DNN-HMM or CTC architectures.

Throughout our experiments, the encoder network consists of 5 Long-Short Term Memory (LSTM) layers each with 1024 hidden units followed by a fully-connect (FC) layer. Similarly, the decoder is comprised of 2 LSTM layers each with 1024 hidden units and a FC layer. The dropout rate for each layer is fixed at 30%. To show the effectiveness and versatility of ACosR for QAT, we considered two languages, i.e. English and French, and conducted multiple experiments with different joint network (JN) architectures as well as numbers of word-pieces (WPs). Please refer to Tab. 1 for descriptions of each scenario. The number of parameters is $\approx 56 - 60$ M for each setup. Our models do not include language model/second-pass rescoring.

4.2. Data

We investigate the model performance for English and French ASR. We train our RNN-T models with 23000 (23k) and 10000 (10k) hours of audio for English and French, respectively, which encompasses all far-field virtual assistant tasks. For testing, we consider three different types of datasets: Frequent, Appliances, and Entertainment. Here, Frequent dataset includes tasks that are frequently queried, while Appliances and Entertainment contain queries from specific supported Appliances and Entertainment tasks. Note that the testing datasets are not included in the training counterpart, and Frequent dataset does not contain either Appliances or Entertainment testset. All datasets are anonymized and human-transcribed. We also train and evaluate an RNN-T model under Setup 1 (see Tab. 1) with Librispeech data [23]. Tab. 2 presents an overview of the amount of training and testing data for various cases.

5. Results and Discussion

In this section, we compare the performance of the models described in Tab. 1 for three cases: floating-point training, in which there is no QAT applied, and QAT via ACosR with either

Table 3: *Comparisons of floating-point baseline with QAT RNN-T models for various test sets. Here, "nWER" represents the WER normalized by that of floating point-Setup1-Entertainment, and "rel. dgrd" denotes relative WER degradation compared to floating-point model in each setup and dataset.*

		Entertainment		Appliances		Frequent		WgtC (%)
		nWER	rel. dgrd. (%)	nWER	rel. dgrd. (%)	nWER	rel. dgrd. (%)	
Setup 1	floating point	1.00	–	0.60	–	1.12	–	–
	8-bit ACosR	0.97	–3.02	0.58	–3.23	1.10	–2.01	99.4
	6-bit ACosR	0.98	–1.19	0.58	–2.51	1.14	–1.05	99.2
Setup 2	floating point	1.04	–	0.58	–	1.12	–	–
	8-bit ACosR	1.02	–1.35	0.58	–0.74	1.12	–0.21	99.5
	6-bit ACosR	1.07	3.12	0.60	2.95	1.14	1.63	99.1
Setup 3	floating point	1.31	–	0.68	–	1.16	–	–
	8-bit ACosR	1.33	1.53	0.69	1.96	1.16	0.00	99.6
	6-bit ACosR	1.34	2.22	0.70	2.37	1.17	1.48	99.0

Table 4: *Actual WERs (%) of floating-point and QAT models for Librispeech data and Setup 1. Largest relative degradations for 8-bit and 6-bit QAT are* 0.66% *and* 2.68%*, respectively.*

	dev-clean	dev-other	test-clean	test-other	WgtC (%)
f. p.	8.11	21.27	8.68	22.29	–
8-bit	8.15	21.41	8.70	22.36	99.3
6-bit	8.32	21.84	8.90	22.82	99.4

8-bit or 6-bit . Here x-bit ACosR means that we apply regularization to achieve a weight distribution with 2^x quantization levels. We fix $\alpha = 0.005$ while $\theta = 2^7$ and $\theta = 2^5$ for 8-bit and 6-bit QAT models, respectively. For far-field datasets, we report the results normalized by the WER for floating point model under Setup 1 and Entertainment dataset, which is smaller than 10%, in Tab. 3. For Librispeech data, we show the actual WER in Tab. 4. From Tab. 3 and Tab. 4, it is observed that the QAT achieves similar performance to the floating point baseline, especially for 8-bit QAT models for which there is zero to little accuracy degradation.

Consider the performance of 8-bit QAT models under far-field datasets first. There is even a slight improvement for 8-bit QAT over floating-point baseline under Setup 1 and 2, which can be explained by the fact that applying regularization leads to better generalization in certain cases. Under Setup 3, however, 8-bit QAT model slightly underperforms its floating-point counterpart. For Librispeech data, we also observe very little degradation, i.e. less than 0.66% relative, for 8-bit QAT. The 6-bit QAT model, on one hand, performs on par with 8-bit QAT and surpasses floating-point baseline under Setup 1 with far-field dataset. On the other hand, under Setup 2 and 3 scenarios or with Librispeech data, 6-bit QAT models achieve consistently worse performance for both far-field and Librispeech datasets. As discussed above, regularization is beneficial under certain cases, however, it cannot mitigate the degradation effect of low-bit quantization on the model performance. Overall, 6-bit QAT models only degrade by at most 5% relative compared to its floating-point cases. The topic of low-bit QAT will be pursued in subsequent works.

In Tab. 3 and Tab. 4, we also report the weight convergence (WgtC), which represents the percentage of model weights that converged to the quantization levels during training. Here a weight is said to be converged to a quantization level if the difference between them is less than 10^{-6}. The convergence rate validates that the vast majority of the weights have converged to one of the quantization levels, thus enabling efficient post-training quantization. To illustrate this point, we further present the distribution for the RNN-T decoder weights under QAT-ACosR during training in Fig. 4. Note that each LSTM layer has three weight components: kernel, recurrent kernel, and bias, while each FC has two: kernel and bias. However, due to the space constraint, here we only show the kernel weights for different decoder layers, and omit other components. The figure has been zoomed in on the near-zero region to clearly depict that the weights are very close to the quantization levels. This will allow efficient post-training quantization of the weights without any significant accuracy loss.

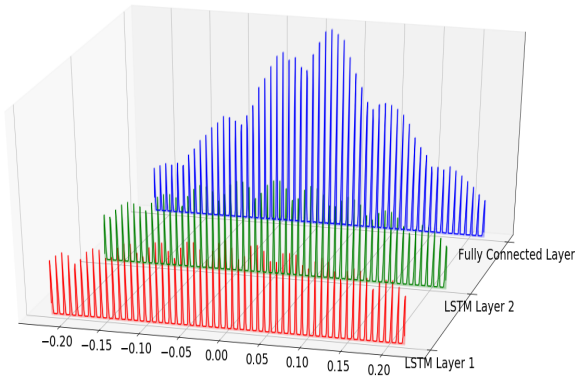

Figure 4: *Decoder kernel weight distribution during training for RNN-T trained with QAT-ACosR.*

6. Conclusions

In this work, we presented a novel quantization-aware training method for deep neural networks using regularization. Our approach imposed to the model a weight distribution which is similar in shape to that of a quantized model. We achieved that by introducing a new function called Absolute-Cosine and incorporating this function to the model loss. Theoretical investigations further confirm that Absolute-Cosine regularization loss is a good approximation for the quantization loss of the model weights. Using only one regularization function, our approach requires much smaller additional memory and computation resources compared to existing QAT schemes. We evaluated our approach in the speech recognition task by utilizing a speech recognition system based on RNN-T. Our results show that QAT with ACosR results into zero to little performance degradation under various test sets.

7. References

[1] R. Alvarez, R. Prabhavalkar, and A. Bakhtin, "On the efficient representation and execution of deep acoustic models," in *Proceedings of the Annual Conference of the International Speech Communication Association (INTERSPEECH)*, 2016, pp. 2746–2750.

[2] B. Jacob, S. Kligys, B. Chen, M. Zhu, M. Tang, A. Howard, H. Adam, and D. Kalenichenko, "Quantization and training of neural networks for efficient integer-arithmetic-only inference," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018, pp. 2704–2713.

[3] W. Chen, J. Wilson, S. Tyree, K. Weinberger, and Y. Chen, "Compressing neural networks with the hashing trick," in *Proceedings of the International Conference on Machine Learning (ICML)*, vol. 37, 2015, pp. 2285–2294.

[4] R. Shu and H. Nakayama, "Compressing word embeddings via deep compositional code learning," in *Proceedings of the International Conference on Learning Representations (ICLR)*, vol. abs/1711.01068, 2018.

[5] M. Georges, K. Czarnowski, and T. Bocklet, "Ultra-compact nlu: Neuronal network binarization as regularization," in *Proceedings of the Annual Conference of the International Speech Communication Association (INTERSPEECH)*, 2019, pp. 809–813.

[6] O. Abdel-Hamid, A. Mohamed, H. Jiang, and G. Penn, "Applying convolutional neural networks concepts to hybrid nn-hmm model for speech recognition," in *Proceedings of the International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2012, pp. 4277–4280.

[7] T. N. Sainath, A. Mohamed, B. Kingsbury, and B. Ramabhadran, "Deep convolutional neural networks for lvcsr," in *Proceedings of the International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2013, pp. 8614–8618.

[8] A. Graves, S. Fernandez, F. Gomez, and J. Schmidhuber, "Connectionist temporal classification: Labelling unsegmented sequence data with recurrent neural networks," in *Proceedings of the International Conference on Machine Learning (ICML)*, 2006.

[9] A. Graves, A. Mohamed, and G. E. Hinton, "Speech recognition with deep recurrent neural networks," in *Proceedings of the International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2013, pp. 6645–6649.

[10] W. Chan, N. Jaitly, Q. Le, , and O. Vinyals, "Listen, attend and spell: A neural network for large vocabulary conversational speech recognition," in *Proceedings of the International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2016, pp. 4960–4964.

[11] A. Graves, "Sequence transduction with recurrent neural networks," in *Proceedings of the International Conference on Machine Learning (ICML)*, 2012.

[12] S. Han, H. Mao, and W. Dally, "Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding," in *Proceedings of the International Conference on Learning Representations (ICLR)*, 2016.

[13] R. Pang, T. Sainath, R. Prabhavalkar, S. Gupta, Y. Wu, S. Zhang, and C.-C. Chiu, "Compression of end-to-end models," in *Proceedings of the Annual Conference of the International Speech Communication Association (INTERSPEECH)*, 2018.

[14] Y. Gong, L. Liu, M. Yang, and L. Bourdev, "Compressing deep convolutional networks using vector quantization," in *arXiv: 1412.6115. URL: http://arxiv.org/abs/1412.6115*, 2014.

[15] E. Denton, W. Zaremba, J. Bruna, Y. LeCun, and R. Fergus, "Exploiting linear structure within convolutional networks for efficient evaluation," in *Proceedings of the 27th International Conference on Neural Information Processing Systems (NIPS)*, 2014, pp. 1269–1277.

[16] M. Jaderberg, A. Vedaldi, and A. Zisserman, "Speeding up convolutional neural networks with low rank expansions," in *Proceedings of the British Machine Vision Conference (BMVC)*, 2014.

[17] Y. Mishchenko, Y. Goren, M. Sun, C. Beauchene, S. Matsoukas, O. Rybakov, and S. N. P. Vitaladevuni, "Low-bit quantization and quantization-aware training for small-footprint keyword spotting," in *Proceedings of the IEEE International Conference On Machine Learning And Applications (ICMLA)*, 2018, pp. 706–711.

[18] Tensorflow, "Quantization-aware training via fake quantization," *https://github.com/tensorflow/tensorflow/tree/r1.14/tensorflow/contrib/quantize*, 2017.

[19] H. Liu, R. Wang, S. Shan, and X. Chen, "Deep supervised hashing for fast image retrieval," in *Proceedings of the Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[20] A. Babenko and V. S. Lempitsky, "Additive quantization for extreme vector compression," in *Proceedings of the Conference on Computer Vision and Pattern Recognition (CVPR)*, 2014.

[21] C. Song, "A quantization-aware regularized learning method in multilevel memristor-based neuromorphic computing system," *Master Thesis*, 2017. [Online]. Available: http://d-scholarship.pitt.edu/31080/7/ChangSong_etdPitt2017.pdf

[22] W. Tang, G. Hua, and L. Wang, "How to train a compact binary neural network with high accuracy?" in *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, 2017, pp. 2625–2631.

[23] V. Panayotov, G. Chen, D. Povey, and S. Khudanpur, "Librispeech: an asr corpus based on public domain audio books," in *Proceedings of the International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2015, pp. 5206–5210.