Interspeech 2022
18-22 September 2022, Incheon, Korea

SQUASHED WEIGHT DISTRIBUTION FOR LOW BIT QUANTIZATION OF DEEP MODELS

Nikko Strom, Haidar Khan, Wael Hamza

Amazon Alexa AI
$\{nikko, khhaida, waelhamz\}$@amazon.com

ABSTRACT

Inference with large deep learning models in resource-constrained settings is increasingly a bottleneck in real-world applications of state-of-the-art AI. Here we address this by low-precision weight quantization. We achieve very low accuracy degradation by re-parameterizing the weights in a way that leaves the weight distribution approximately uniform. We show lower bit-width quantization and less accuracy degradation than previously reported in experiments on GLUE benchmarks (3-bit, 0.2% rel. degradation), and on internal intent/slot-filling datasets (2-bit, 0.4% rel. degradation).
Index Terms: quantization, deep learning, transformers

1. INTRODUCTION

Very large machine learning models, with a high count of learnable parameters is a hallmark of contemporary deep learning. A number of studies have shown and quantified that in many settings, larger is better, e.g., [1]. In particular for Natural Language Processing (NLP), large transformer-based language models trained on terrabytes of unlabeled data have recently formed the backbone for systems that excel on important benchmarks [2, 3]. As the success and size of such models grow, so too does the importance of compressing them without sacrificing too much of their performance. Compression is motivated by inference speed/cost, for enabling deployment in resource-constrained edge devices, and a lower carbon footprint [4, 5].

Techniques for compressing neural network models range from pruning weights [6, 7], to low-rank factorization [8], to training a small model under the supervision of a larger one [9]. One method with a rich history is quantizing the model parameters into fewer bits [10, 11]. Quantization is a straight-forward approach for reducing size and increasing speed and, with bit-widths of eight bits or more, it can often be done easily with very little degradation in accuracy [12]. However for a given model and task setting, each quantization method has a characteristic accuracy versus compression curve where accuracy typically degrades rapidly above some level of aggressive quantization. For transformer models in particular, it has been shown difficult to quantize weights to fewer than four bits without significant degradation [13, 14] and many works focus on 8-bit quantization, e.g., [15].

A basic approach is full-precision training followed by quantization, but for aggressive compression the quantization noise introduced by this mismatch between training and test can degrade accuracy. This can be mitigated by quantization-aware training, [11], and more recently [12], which introduces quantization during training, typically by injecting quantization noise in the forward pass, and estimating the gradients of quantized parameters with Straight Through Estimators (STE) [16]. Some studies have pointed out that as a secondary effect, STE introduces a small bias during training and have proposed extensions to the method [17].

A relatively unexplored approach to quantization-aware training is studying and optimizing the overall distribution of weights under the typical stochastic gradient descent (SGD) optimization. In this paper we take this approach when we consider the problem of quantizing large transformer-based language models into low bit-widths (sub 4-bit precision). This type of compression level promises to reduce memory footprint even further and allow very fast inference by leveraging low bit precision operators.

Our contributions We propose (i) a novel weight transformation that causes SGD to learn approximately uniformly distributed weights instead of the typical Gaussian distribution. Further, we apply (ii) a regularization technique that controls the mean and variance of the learned parameters, and we show how it can be efficiently implemented in modern deep learning frameworks. Finally, (iii) we demonstrate empirically, as far as we know, the lowest reported quantization bit-widths for compressed transformer models with only minor accuracy degradations measured on common benchmark tests - only 0.2% relative degradation with 3-bit precision.

2. BACKGROUND

Given a vector of weights w, each individual weight w_i is linearly mapped to a finite range, typically $[-1,1]$, by normalizing: $\hat{w_i} = 2\frac{w_i - \min(w)}{\max(w)-\min(w)} - 1$ where $\min(w)$ and $\max(w)$ are stored separately for each weight field. To quantize the weights, 2^b points are spaced uniformly $[-1,1]$ and $\hat{w_i}$ is mapped to the closest point:

$$Q(w_i) = \Big(\text{round}[2^{b-1}(\hat{w_i}+1) - 0.5] + 0.5\Big) * 2^{1-b} - 1. \quad (1)$$

With the quantization aware training approach, the quantized parameters are directly optimized with STE [16], since the $\text{round}[\cdot]$ operation is non-differentiable.

Outliers in the distribution of the unquantized weights stretch the weight range and wider ranges space the quantized points further apart, increasing the quantization error: $|w_i - Q(w_i)|$. This can be mitigated by approaches that attempt to adapt to the natural bell-curve of the weight distribution (discussed in Section 6). We adopt a different approach, shaping the weight distribution so that it resembles the uniform distribution where each quantization level is utilized equally.

Our method can be applied to any weight fields, but in the rest of this paper, we focus on the transformer neural network architecture [18]. The model-weights of the transformer are contained in affine linear operations that transform an input x by $Wx + b$ where W is a matrix of weights and b is a vector of bias terms. We focus on quantizing the weight matrices W that form the bulk of the size and computation in the network.

Copyright © 2022 ISCA

10.21437/Interspeech.2022-50

3. METHODS

Weights in neural networks optimized with stochastic gradient descent are generated by aggregating a large number of stochastic updates and as understood by the central limit theorem, this typically results in Gaussian distributed weights. For example, this is observed empirically in the weights of Transformer models [18] and popular convolutional neural networks like ResNet [19]. However, for quantizing weights, the Gaussian distribution is not particularly efficient. Individual weights in the tails of the bell-curve are rare, but we still must allocate precious bits to them.

Quantization would be most efficient if weights were uniformly distributed, for example $\mathcal{U}(-1,1)$. This leads us to the key idea to make our model weights a function f of the actual SGD-trained parameters P. Specifically, we want to select f such that if $P \sim \mathcal{N}(0,\sigma^2)$, then $W = f(P) \sim \mathcal{U}(-1,1)$. By definition, we can use the Gaussian cumulative distribution function $f(p) = 2\,\mathrm{erf}(p) - 1$, to achieve this, and this is a feasible variant of the method, but for some reduction in computation, here we opt to use the rough approximation $f = \tanh(\cdot)$. In the following, we derive the distribution of $\tanh(P)$ when $P \sim \mathcal{N}(0,\sigma^2)$.

Given $P \sim \mathcal{N}(0,\sigma^2)$, $W = \tanh(P)$, $p = \operatorname{arctanh}(w)$ with $w \in [-1,1]$ we have $\mathbb{P}(W \le w) = \mathbb{P}(\operatorname{arctanh}(W) \le p)$. Further:

$$\mathbb{P}(\operatorname{arctanh}(W) \le w) = \frac{1}{\sqrt{2\pi}\sigma}\int_{-\infty}^{\operatorname{arctanh}(w)} e^{-\frac{\xi^2}{2\sigma^2}}\,d\xi \qquad (2)$$

$$\frac{d\mathbb{P}(W \le w)}{d\operatorname{arctanh}(w)} = \frac{1}{\sqrt{2\pi}\sigma}e^{-\frac{\operatorname{arctanh}^2(w)}{2\sigma^2}} \qquad (3)$$

$$W \sim \frac{d\mathbb{P}(W \le w)}{dw} = \frac{d\mathbb{P}(W \le w)}{d\operatorname{arctanh}(w)}\frac{d\operatorname{arctanh}(w)}{dw} \qquad (4)$$

$$W \sim \frac{1}{\sqrt{2\pi}\sigma(1-w^2)}e^{-\frac{\operatorname{arctanh}^2(w)}{2\sigma^2}} \qquad (5)$$

Not surprisingly, the shape of the distribution of W depends on the variance σ^2. This can be treated as a hyperparameter, but intuitively, we want the transformed distribution to resemble $\mathcal{U}(-1,1)$. To that end, in Figure 1, we plot the distribution for σ optimized with a few different similarity objectives, and from visual inspection of Figure 1, we select the target $\sigma_t = 0.8 \approx \sqrt{2/\pi}$ in our experiments. Note: if we had used the exact $f(p) = 2\,erf(p) - 1$ we would simply use the target $\sigma_t = 1.0$.

To achieve this target weight distribution, we first initialize weights with the desired distribution, $\mathcal{N}(0,\sigma_t^2)$, and then we train with a regularization loss L_Q that penalizes deviations from the target distribution:

$$L_Q = \lambda_q\left((\operatorname{stddev}(P) - \sigma_t)^2 + \operatorname{mean}(P)^2\right). \qquad (6)$$

This regularization encourages W to be uniformly distributed. Our initialization insures that L_Q is initially zero, and since the mean and variance statistics change slowly compared to changes in individual weights, we can use a low rate λ_q on the regularization term while still maintaining the target distribution throughout the training. As shown in (6), we rely on auto-differentiating stddev and mean, but it can be shown that the gradient can also be formulated where this is not required:

$$\frac{dL_Q}{dp_i} = \lambda_q\left(2\rho p_i + \mu\right), \quad \begin{array}{c}\rho = \frac{\operatorname{stddev}(P)-\sigma}{\operatorname{stddev}(P)} \\ \mu = 2\operatorname{mean}(P)(1-\rho)\end{array} \qquad (7)$$

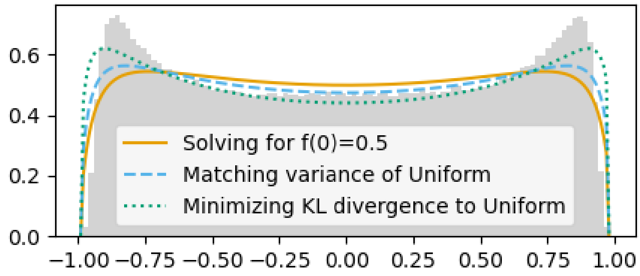

Fig. 1. Resulting weight distribution derived by optimizing σ_t in three ways. If we match the variance of the distributions, we get the optimum for $\sigma_t = 0.8424$. Minimizing the KL divergence yields $\sigma_t = 0.9069$ (the exact value is $\pi/\sqrt{12}$). However, in our experiments, we use $\sigma_t = 0.8$, which approximately corresponds to solving $f(0) = 0.5$ (solid orange line). The exact value is $\sqrt{2/\pi}$. The shaded region represents the normalized histogram of weights in a network trained with $\sigma_t = 0.8$. This empirical distribution does not exactly coincide with the predicted as our Gaussian assumption is approximate. Nonetheless, the resulting distribution makes linear quantization very information efficient.

Thus far we have shown that we can re-parameterize the weights W as $f(P)$ to change the distribution by applying a squashing function (tanh). However, applying the squashing function to the weights restricts the range of the output. We deal with this by introducing a vector of gain parameters g (the same size as b) and apply them to the output to re-enable the full range: $\tanh(P)xe^g + b$. We initialize e^g to achieve an initial weight distribution that is commensurable with best practices for training models of a certain architecture. For our experiments, inspired by Xavier initialization [20], we used $g_{init} = -\log(\sigma_t^2\sqrt{n_{in}})$, where n_{in} is the number of input features to the linear transform.

Because SGD is not scale-invariant, transforming the parameters impacts the optimal learning rate for training. Empirically we found that with our method, a good choice for learning rate is about 10x higher than for the original model training. This can be explained by the fact that the gain e^g and the ≤ 1 magnitude of the derivative of tanh act to scale down the gradient of the parameters P.

In summary, we modify the original transformer architecture by replacing the linear operation with $tanh(P)xe^g$ and train the parameters P with the quantization regularization loss L_Q. For quantization-aware training, we use STE [16] to estimate the gradients by ignoring non-differentiable operators in the backward pass. Note that quantization is applied to $\tanh(P)$ – not to P. When training is completed, we can convert the model back to the original transformer architecture, but with aggressively quantized weights.

4. RESULTS

We evaluate our quantization method, Squashed Weight Quantization (SqWQ), on three tasks: language modeling, sentence classification, and token classification.

4.1. Language modeling

In this set of experiments, we show that the architectural changes and regularization (without quantization) we introduce with SqWQ do not degrade performance on unsupervised language modeling tasks.

Model	Teacher	Objective	Perplexity
BERT	-	MLM only	7.97
DistilBERT	BERT	Distilled	8.65
leanBERT	BERT	Distilled	8.71
SqWQ BERT	-	MLM only	9.12
SqWQ BERT	BERT	Distilled	7.85
SqWQ DistilBERT	BERT	Distilled	8.51
SqWQ leanBERT	BERT	Distilled	8.95

Table 1. Perplexity comparison of models trained with our proposed architecture and loss function (SqWQ*) to unmodified baseline models. DistilBERT and leanBERT are 6 and 4 layer BERT variants.

Pretraining We pretrain transformer models with SqWQ using the masked language modeling (MLM) objective described in [21] with modifications proposed in [2]. Namely, we drop the sentence level tasks like next sentence prediction (NSP) and truncate sentences to 256 characters to reduce training time. We perform pretraining on the Wikipedia and the Google Billion Word [22] datasets.

The results on a held out test set are presented in Table 1. The gap between the standard BERT and SqWQ BERT is about 1 perplexity point. In the following experiments we show how we can close this gap with distillation.

Distillation We perform distillation using two variants proposed in the literature on the MLM task: MLM loss combined with cosine distance based distillation losses [23] and MLM loss with mean squared error based distillation losses [24].

Table 1 shows the results of distillation with SqWQ. We first show that it is possible to close the perplexity gap between standard BERT and SqWQ BERT by incorporating the distillation losses. Using the standard BERT model as a teacher, we show that we can match the perplexity between the 6-layer DistilBERT and SqWQ DistilBERT. The approach also holds for the shorter 4-layer transformer (leanBERT) where the standard leanBERT and SqWQ leanBERT perplexity are nearly equal.

4.2. Natural language understanding

We finetune the distilled SqWQ models and compare them to strong baselines on several sentence level and token level language understanding tasks.

Sentence level classification We consider 6 tasks from the GLUE set of benchmark tasks: MNLI, MRPC, QNLI, QQP, RTE, and SST-2 [25]. Results on the dev sets of these tasks are shown in Table 2, where we compare 3 models: uncompressed models, models quantized with iterative Product Quantization (iPQ) [26], and models with SqWQ.

Token level classification Our models are also evaluated on tasks involving token level decisions, specifically the joint intent classification and slot labeling (ICSL) task. For this task, the model outputs a prediction for each input token in the sentence as well as an intent for the entire sentence. For example:

MOVIEINTENT: $\underbrace{\text{play}}_{\text{Other}}\ \underbrace{\text{i}}_{\text{Movie}}\ \underbrace{\text{robot}}_{\text{Movie}}\ \underbrace{\text{on}}_{\text{Other}}\ \underbrace{\text{my}}_{\text{Other}}\ \underbrace{\text{tv}}_{\text{Device}}$

This internal dataset (ICSL) consists of 26 domains with a combined total of 40M training, 8M validation, and 888K test examples.

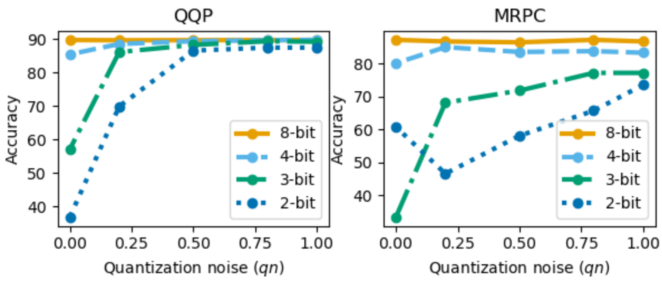

Fig. 2. Results using SqWQ DistilBERT on the dev sets of two GLUE tasks with the combination of QuantNoise [17] and our proposed method. qn=1.0 corresponds to the standard quantization aware training setting while qn=0.0 is training without quantizing any parameters in the forward pass.

The data is unevenly split between domains, with 7 high resource domains and 19 medium-low resource domains.

We compare the performance of iterative product quantization (iPQ) and our method at approximately the same compression level on this task. We measure the Semantic Error (SemER), defined by $SemER = \frac{D+I+S}{N}$. where C, D, I, and S represent the count of correct, deleted, inserted, and substituted slots, where intent is counted as a special slot, and N is the total reference count of slots plus one for the intent.

Table 3 shows the weighted average performance over 26 domains for the uncompressed model, iPQ with varying block size, and SqWQ with 8, 4, 2, 1-bit precisions. Our method maintains performance relative to the uncompressed baseline at much higher compression ratios than iPQ. Even at 1-bit precision, models trained with SqWQ show very small degradation relative to the baseline (0.42%).

5. ANALYSIS

In this section, we study the decision of when to quantize the network explicitly (pretraining vs finetuning), combine our method with a quantization aware training approach such as QuantNoise [17], and compare to post training quantization.

Fixing quantization levels Our proposed method permits the flexibility to choose the quantization level for the model at pretraining time or finetuning time. In Table 4, we show that there is little degradation associated with pretraining without explicitly choosing the number of quantization levels and deferring the actual quantization to finetuning time.

QuantNoise and SqWQ To study the effect of bias introduced by the STE, we combined our method with QuantNoise [17]. QuantNoise deals with STE biased gradients by quantizing only a random subset of weights in each forward pass. This allows unbiased gradients to flow through the unquantized weights and was demonstrated to improve performance over standard quantization aware training [17]. We apply QuantNoise to SqWQ and present the results in Figure 2. We show that in some cases combining QuantNoise with SqWQ can slightly improve performance. However in the low bit regime of interest (4 bits and below), the effect is not observed. In addition, we conclude that with our approach standard quantization aware training (qn=1.0) is the most effective.

Method	Comp/Bits	MNLI	MRPC	QNLI	QQP	RTE	SST2	Average
No Quant	1.0/32 bit	82.52	84.31	87.13	90.08	59.57	92.09	82.56
iPQ	5.01/-	81.78	83.09	86.44	89.49	55.6	91.4	81.31
SqWQ	3.97/8 bit	81.5	83.58	87.41	89.72	58.84	92.32	82.15
SqWQ	7.88/4 bit	80.59	82.6	86.51	88.74	59.21	91.74	81.43
SqWQ	12.00/3 bit	81.66	83.82	86.34	89.37	62.09	92.2	82.39
SqWQ	15.51/2 bit	79.72	73.77	84.44	88.73	58.12	90.71	79.36
SqWQ	30.06/1 bit	71.57	72.3	69.98	85.74	53.07	83.37	72.53

Table 2. Accuracy on the development sets of 6 GLUE tasks with the DistilBERT architecture (3 seed median). Our method outperforms iPQ (b=2) at higher compression ratios. In the best setting (3-bit SqWQ), we achieve 12x compression at only 0.2% relative degradation.

Model	Rel. SemER	Comp. Ratio
leanBERT	0.0 %	1.0
iPQ leanBERT (b=1)	0.0 %	2.65
iPQ leanBERT (b=2)	0.11 %	3.66
iPQ leanBERT (b=4)	2.53 %	4.49
8-bit SqWQ leanBERT	0.0 %	3.96
4-bit SqWQ leanBERT	0.11 %	7.83
2-bit SqWQ leanBERT	0.42 %	15.29
1-bit SqWQ leanBERT	0.42 %	29.23

Table 3. Relative SemER and compression ratio on ICSL of iPQ versus SqWQ. SqWQ enables much higher compression ratios with small performance degradation.

Quant @ Distill	Quant @ FT	Shop	Music	Comm.
4 bit	4 bit	**0.51** %	**0.05** %	-0.30 %
None (32 bit)	4 bit	0.68 %	0.08 %	**-0.36** %
3 bit	3 bit	**0.60** %	**0.39** %	**-0.22** %
None (32 bit)	3 bit	0.68 %	0.51 %	-0.18 %

Table 4. Comparing quantization during both distillation and finetuning to only during finetuning on ICSL. We report SemER relative to the unquantized model (leanBERT) where ↓ is better. There is a small loss in performance associated with deferring quantization to finetuning, but the benefit is that quantization level is flexible during finetuning.

SqWQ vs post-training quantization The simplest baseline for quantization aware methods like SqWQ is post-training quantization. This corrsponds to qn=0.0 in Figure 2 where we see that performance degrades significantly for low bit widths.

6. RELATED WORK

Quantization of neural networks has attracted a large body of work over the last few decades. A special case is binary networks (1 bit), where [27, 28, 29] are early works. In [30] the authors demonstrated quantized backpropagation with ternary weights and specialized multiplication hardware to reduce both training and inference time. Other approaches have targeted higher bitwidths, such as 4-bit [31] and 8-bit [15]. Besides scalar quantization, vector quantization, such as iPQ [26], group weights into clusters and represent weights by combinations of cluster centroids via codebooks. Typically, the vector quantization approach yields higher compression ratios than scalar quantization but is less straighforward to leverage without specialized operators.

To close the performance gap to unquantized models, nearly all quantization approaches train the network with quantization operations and the straight through estimator [16]. Closing this gap at low bitwidths is challenging, for example, with quantization interval training [31] the performance degradation relative to the unquantized baseline is still $> 5\%$. Even with learnable quantizers [32] the performance drop is non-negligible. Our approach aims to close this gap further for low bitwidths.

The long-tailed Gaussian weight distribution has been identified as a challenge for effective quantization [33]. Clipping outliers and non-uniform quantization [34, 35] are straightforward approaches to deal with the long tail. By adding redundant connections to the network, [33] iteratively split the weights in the tail to reduce the number of outliers. Our work tackles this problem from a different perspective, directly addressing the distribution of the weights during training.

There has been recent efforts to quantize transformer models for machine translation [13], language modeling [15, 14], and natural language understanding [17]. Several of these works rely purely on STE for quantization aware training (QAT) and achieve promising results [13, 15]. The basic QAT approach is improved by adding quantization noise during training [17] and jointly optimizing quantized weights and variable bitwidths [14]. However, SqWQ compares favorably with these methods at low bitwidths.

7. CONCLUSION

Our approach to quantizing large neural networks allows compression to very small bit-widths without significant performance degradation. We achieve this with a simple re-parameterization of the weights that squash the distribution, and by introducing a regularization term to the training loss. Our results are shown to hold across common unsupervised and supervised NLP benchmarks and for state-of-the-art transformer models, and we will report on applying this general method to other architectures in the future. Concurrently, our technique has been applied to keyword spotting models [36].

We intend to explore composing our approach with other quantization approaches, such as adapting absolute cosine regularization [37] for transformers. One interesting direction is enabling variable bitwidths across model parameter groups [14] that can further close the gap between the quantized and unquantized models. We are also interested in applying our approach to activations to further improve inference latency.

8. ACKNOWLEDGEMENTS

We thank Hari Parthasarathi and Stephen Rawls for insightful comments and discussion.

9. REFERENCES

[1] T. Brown *et al.*, "Language models are few-shot learners," in *Advances Neural Inf. Process. Syst.*, H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin, Eds., vol. 33. Curran Associates, Inc., 2020, pp. 1877–1901.

[2] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, and V. Stoyanov, "RoBERTa: A robustly optimized BERT pretraining approach," *arXiv*, vol. abs/1907.11692, 2019.

[3] M. Lewis, Y. Liu, N. Goyal, M. Ghazvininejad, A. Mohamed, O. Levy, V. Stoyanov, and L. Zettlemoyer, "BART: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension," in *Proc. 58th Annu. Meeting Assoc. Comput. Linguistics*. Assoc. Comput. Linguistics, Jul. 2020.

[4] E. Strubell, A. Ganesh, and A. McCallum, "Energy and policy considerations for deep learning in NLP," *arXiv*, vol. abs/1906.02243, 2019.

[5] E. M. Bender, T. Gebru, A. McMillan-Major, and S. Shmitchell, "On the dangers of stochastic parrots: Can language models be too big?" in *Proc. 2021 ACM Conf. Fairness, Accountability, and Transparency*, 2021.

[6] Y. LeCun, J. S. Denker, and S. A. Solla, "Optimal brain damage," in *Advances Neural Inform. Process. Syst.*, 1990.

[7] N. Ström, "Sparse connection and pruning in large dynamic artificial neural networks," in *Fifth European Conf. Speech Commun. and Technology*, 1997.

[8] 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 *2013 IEEE Intl. Conf. Acoustics, Speech and Signal Processing*, 2013.

[9] G. Hinton, O. Vinyals, and J. Dean, "Distilling the knowledge in a neural network," *arXiv*, vol. abs/1503.02531, 2015.

[10] W. Balzer, M. Takahashi, J. Ohta, and K. Kyuma, "Weight quantization in Boltzmann machines," *Neural Networks*, vol. 4, no. 3, pp. 405–409, 1991.

[11] D. Orrey, D. Myers, and J. Vincent, "A high performance digital processor for implementing large artificial neural networks," in *Proc. IEEE 1991 Custom Integrated Circuits Conf.* IEEE, 1991, pp. 16–3.

[12] 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 *Proc. IEEE Conf. Comput. Vision and Pattern Recognition*, 2018.

[13] G. Prato, E. Charlaix, and M. Rezagholizadeh, "Fully quantized transformer for machine translation," in *Findings of the Assoc. for Comput. Linguistics: EMNLP 2020*. Assoc. for Comput. Linguistics, Nov. 2020, pp. 1–14.

[14] A. Défossez, Y. Adi, and G. Synnaeve, "Differentiable model compression via pseudo quantization noise," *arXiv*, vol. abs/2104.09987, 2021.

[15] O. Zafrir, G. Boudoukh, P. Izsak, and M. Wasserblat, "Q8BERT: Quantized 8bit BERT," *arXiv*, vol. abs/1910.06188, 2019.

[16] Y. Bengio, N. Léonard, and A. C. Courville, "Estimating or propagating gradients through stochastic neurons for conditional computation," *arXiv*, vol. abs/1308.3432, 2013.

[17] P. Stock, A. Fan, B. Graham, E. Grave, R. Gribonval, H. Jegou, and A. Joulin, "Training with quantization noise for extreme model compression," in *Int. Conf. Learning Representations*, 2021.

[18] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. u. Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances Neural Inform. Process. Syst.*, 2017, pp. 5998–6008.

[19] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *2016 IEEE Conf. Comput. Vision and Pattern Recognition*, 2016.

[20] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," in *Proc. 13th Intl. Conf. Artificial Intell. and Stat*. JMLR Workshop and Conference Proceedings, 2010.

[21] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of deep bidirectional transformers for language understanding," in *Proc. 2019 Conf. North American Chapter Assoc. Computational Linguistics: Human Language Technologies*, 2019.

[22] C. Chelba, T. Mikolov, M. Schuster, Q. Ge, T. Brants, P. Koehn, and T. Robinson, "One billion word benchmark for measuring progress in statistical language modeling," in *15th Annual Conf. Intl. Speech Communication Assoc.*, 2014.

[23] V. Sanh, L. Debut, J. Chaumond, and T. Wolf, "DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter," in *NeurIPS EMC2 Workshop*, 2019.

[24] X. Jiao, Y. Yin, L. Shang, X. Jiang, X. Chen, L. Li, F. Wang, and Q. Liu, "TinyBERT: Distilling BERT for natural language understanding," in *Findings Assoc. Comp. Linguistics: EMNLP*. Online: Association for Computational Linguistics, Nov. 2020.

[25] A. Wang, A. Singh, J. Michael, F. Hill, O. Levy, and S. R. Bowman, "GLUE: A multi-task benchmark and analysis platform for natural language understanding," in *Intl. Conf. Learning Representations*, 2019.

[26] P. Stock, A. Joulin, R. Gribonval, B. Graham, and H. Jégou, "And the bit goes down: Revisiting the quantization of neural networks," in *Intl. Conf. on Learning Representations*, 2020.

[27] M. Courbariaux, Y. Bengio, and J.-P. David, "Binaryconnect: Training deep neural networks with binary weights during propagations," in *Advances Neural Inform. Process. Syst.*, 2015, pp. 3123–3131.

[28] I. Hubara, M. Courbariaux, D. Soudry, R. El-Yaniv, and Y. Bengio, "Binarized neural networks," in *Advances Neural Inf. Process. Syst.*, D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, and R. Garnett, Eds., vol. 29. Curran Associates, Inc., 2016.

[29] M. Rastegari, V. Ordonez, J. Redmon, and A. Farhadi, "Xnor-net: Imagenet classification using binary convolutional neural networks," in *European Conf. Comput. Vision*. Springer, 2016, pp. 525–542.

[30] Z. Lin, M. Courbariaux, R. Memisevic, and Y. Bengio, "Neural networks with few multiplications," in *Intl. Conf. Learning Representations*, 2016.

[31] S. Jung, C. Son, S. Lee, J. Son, Y. Kwak, J.-J. Han, S. J. Hwang, and C. Choi, "Learning to quantize deep networks by optimizing quantization intervals with task loss," in *Proc. IEEE/CVF Conf. Comput. Vision and Pattern Recognition*, 2019, pp. 4350–4359.

[32] D. Zhang, J. Yang, D. Ye, and G. Hua, "LQ-nets: Learned quantization for highly accurate and compact deep neural networks," in *Proc. European Conf. Comput. Vision*, 2018, pp. 365–382.

[33] R. Zhao, Y. Hu, J. Dotzel, C. De Sa, and Z. Zhang, "Improving neural network quantization without retraining using outlier channel splitting," in *Intl. Conf. Machine Learning*. PMLR, 2019.

[34] S. Shin, K. Hwang, and W. Sung, "Fixed-point performance analysis of recurrent neural networks," in *2016 IEEE ICASSP*. IEEE Press, 2016.

[35] H. Yu, T. Wen, G. Cheng, J. Sun, Q. Han, and J. Shi, "Low-bit quantization needs good distribution," in *2020 IEEE/CVF Conf. on Comput. Vision and Pattern Recognition Workshops*, 2020, pp. 2909–2918.

[36] L. Zeng, S. H. K. Parthasarathi, Y. Liu, A. Escott, S. Cheekatmalla, N. Strom, and S. Vitaladevuni, "Sub 8-bit quantization of streaming keyword spotting models for embedded chipsets," in *Text, Speech, and Dialogue*, 2022.

[37] H. D. Nguyen, A. Alexandridis, and A. Mouchtaris, "Quantization aware training with absolute-cosine regularization for automatic speech recognition," in *INTERSPEECH*, 2020.