Quantized Transformer

Chaofei Fan
Stanford University
stfan@stanford.edu

Abstract

Transformer is a powerful architecture for various NLP tasks. However, it requires significant amount of computation and memory resources. In this work, we study how quantized transformer perform on machine translation, sentence classification, and question answering tasks. We show that transformer can be quantized to 8 bits with little loss of performance. When training data is scarce, 8 bits quantization can act as regularizer and improve performance. 8 bits quantized model is 4x smaller than 32 bits model. This will enable deployment of large transformer model on server. With optimized INT8 matrix kernel, 8 bits quantized transformer can also run efficiently on mobile devices, making it possible to translate offline.

1 Introduction

Recently transformer [1] architecture has become the backbone of many state of the art (SOTA) models [2][3][4]. The major novelty of transformer is self-attention, allowing tokens in sequence to interact with each other and also making efficient use of of GPU. However, the model size of all the SOTA transformer models are big, with number of parameters ranging from 300M to 1.5B [2][4]. Deploying these models on servers requires a significant amount of memory ($\sim$3.7GB for BERT large model). And it is almost impossible to run them on edge devices like smart phones.

One widely used method to reduce model size is quantization. Quantization works by representing the full precision (32 bits) model weights with less bits. Quantization not only reduces model size but also improves energy efficiency. For example, with 8 bits integer quantization, model uses 4x less memory and 18-30x less energy [5]. Previous works have shown that quantized model can achieve comparable performance with full precision ones in image classification tasks [6][7][8][9]. In NLP, [10] has shown that word embeddings can be quantized to 1 or 2 bits and still performs as good as the full precision ones.

Inspired by these works, we study how a quantized transformer model perform on machine translation, sentence classification, and question answering tasks. We apply range based linear quantization and binary quantization during training to the original transformer model [1] and the SOTA BERT [2] model. Our results show the followings:

- Initialize quantized model from the pretrained full precision model reduces training time and improves performance compared to train quantized model from scratch.
- 8 bits quantized model performs almost as good as full precision one. In cases where training data is scarce, 8 bits quantization can act as regularizer and outperform full precision one. However, more aggressive quantization such as 1 and 4 bits hurts performance a lot.
- Weight only quantization shows promising results compared to quantizing both weights and activations. If model size is the only concern, it may be possible to train an extra large model and apply aggressive weight only quantization.

2 Related Work

Neural network quantization is a hot area of research. Most studies focus on two types of quantization: 8 bits and 1 bit. 8 bits quantization is the most practical method. It uses INT8 to represent weights and activations, reducing memory usage by 4x. With optimized INT8 matrix kernel, 8 bits quantized model can run efficiently on mobile devices with little loss of accuracy[11][8]. 1 bit quantization reduces model size and computation more dramatically at the cost of higher accuracy loss [7][12]. However, one work shows that word embeddings can be quantized to 1 bit and achieves higher accuracy than full precision model [10].

There are other model compression techniques such as pruning [13] and knowledge distillation [14]. Pruning takes pretrained model and removes weights according to certain criteria. Knowledge distillation trains a small model to mimic a large model. These methods are orthogonal to quantization.

To the best of our knowledge, there is no work studying quantizing transformer model to below 16 bits. The majority of researches on transformer is to create ever larger models to study their limits.

3 Approach

3.1 Background

3.1.1 Transformer

Transformer is a neural network architecture proposed in [1] for machine translation. Its uses self-attention instead of RNN to process sequences of data. Self-attention allows each token in the sequence to attend all other tokens, eliminating the vanishing gradient problem of RNN.

The major building block of transformer includes a multi-head self-attention layer and a point-wise feed forward layer. The input to the building block consists of l tokens of dimension d_{model}. A self-attention head projects each token vector to a set of query, key, and value vectors and uses scaled dot product to compare and combine them:

$$\begin{aligned} Q = XW_Q \quad K = XW_K \quad V = XW_V \\ Attention(Q, K, V) = softmax(\frac{QK^T}{\sqrt{d_k}})V \end{aligned} \tag{1}$$

where $W_Q \in \mathbb{R}^{d_{model} \times d_k}$, $W_K \in \mathbb{R}^{d_{model} \times d_k}$, and $W_V \in \mathbb{R}^{d_{model} \times d_v}$ are projection matrices for query, key, and values, $X \in \mathbb{R}^{l \times d_{model}}$ the input matrix for sequence of length l. Multi-head self-attention is the concatenation of multiple self-attention heads with a linear projection:

$$\begin{aligned} MultiHead(X, X, X) = Concat(head_1, ..., head_h)W_O \\ head_i = Attention(XW_Q, XW_K, XW_V) \end{aligned} \tag{2}$$

where h is the number of attention heads and $W_O \in \mathbb{R}^{hd_v \times d_{model}}$ is the linear projection matrix. Point-wise feed forward layer consists of two linear layers with ReLU in between. It is applied to each input token individually:

$$FFN(x) = ReLU(XW_1 + b_1)W_2 + b_2 \tag{3}$$

where $W_1 \in \mathbb{R}^{d_{model} \times d_{ff}}$, $W_2 \in \mathbb{R}^{d_{ff} \times d_{model}}$, $b_1 \in \mathbb{R}^{1 \times d_{ff}}$, $b_2 \in \mathbb{R}^{1 \times d_{model}}$ and d_{ff} is the dimension of of the first layer. Both multi-head self-attention layer and point-wise feed forward layer are followed by layer normalization and residual connection.

Multiple such building blocks are stacked together to form the transformer model. Because self-attention losses the positional information in the sequence, a positional encoding is added to the embedding.

3.1.2 Quantization

Quantization is a process to reduce the number of bits representing a number. Neural networks by default uses 32 bits floating number as its weights and activations. Both of them can be quantized. Quantization can be applied to a trained model or integrated into the training process. We will

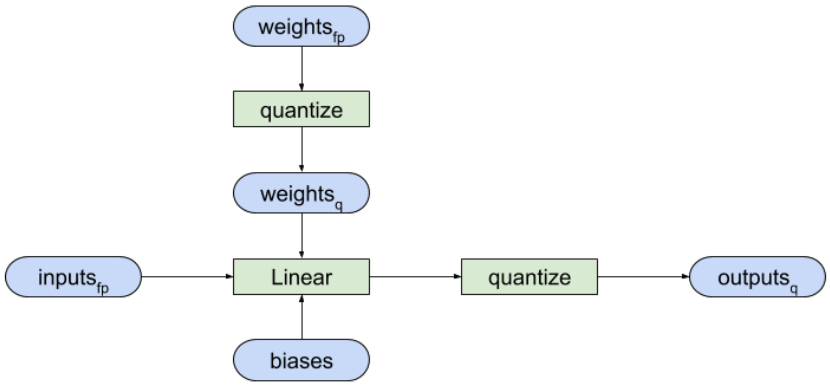

Figure 1: Linear layer quantization during training.

focus on training with quantization since it generally gives better performance than post training quantization.

Figure 1 shows an example of quantized linear layer. A $quantize$ operation is applied to both the layer's weights and outputs to obtain the quantized $weights_q$ and $outputs_q$. We use range based linear $quantize$ operation, which rescales the full precision float number to the range that is representable by the n bits integer. It can be defined as follow:

$$quantize(W_{fp}, n) = round((W_{fp} - min(W_{fp}))\frac{2^n - 1}{max(W_{fp}) - min(W_{fp})}) \tag{4}$$

where W_{fp} is the full precision matrix and n is the number of bits to quantize to.

A special binary form of quantization operation is proposed in [7] where the weights and activations are deterministically mapped to -1 and 1. The $binarize$ operator is defined as follow:

$$binarize(W_{fp}) = sign(W_{fp}) \tag{5}$$

We empirically find that this form of 1 bit quantization performs better than range based linear quantization with $n = 1$. We will use $binarize$ for 1 bit quantize in this work.

Both $quantize$ and $binarize$ operators are not differentiable. To backpropgate through them, we use the straight-through estimator (STE) proposed in [15]. STE approximates the gradient with an identity function:

$$\frac{\partial quantize}{\partial W} = \mathbb{1} \tag{6}$$

$$\frac{\partial binarize}{\partial W} = \mathbb{1}_{|W| \leq 1} \tag{7}$$

Note that in the $binarize$ case, the gradient is passed through only if W is within range $[-1, 1]$. This helps training as found in [7][6].

During training, full precision weights are still needed to get correct estimate of gradients. After training, models can be saved with quantized weights to save storage.

3.2 Quantized Transformer

We apply 1, 4, 8 bits quantization to the transformer model. For 4 and 8 bits quantization, Equation 4 is applied to the weights of multi-head self-attention, point-wise feed forward, and embedding layers. Biases are unchanged because they takes very small amount of storage.

For 1 bit quantization, we add a temperature parameter τ to Equation 1:

$$Attention(Q, K, V) = softmax(\frac{QK^T}{\tau\sqrt{d_k}})V \tag{8}$$

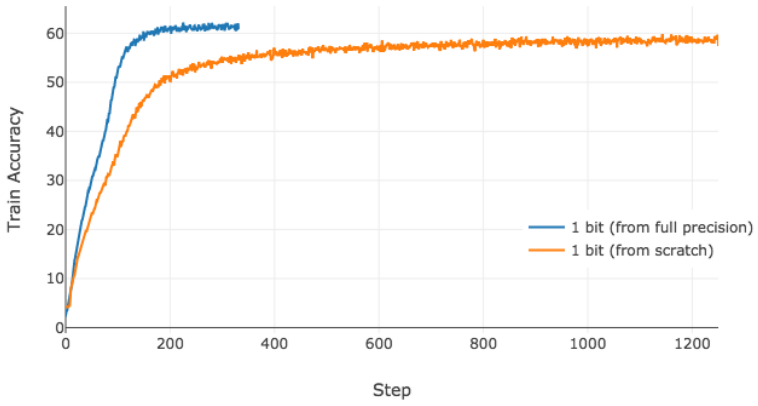

Figure 2: Binary weight model initialized with pretrained full precision model reaches top performance much faster than trained from scratch.

Without this parameter, training loss plateaus at very value. We experiment setting this value to 100, 200, 300, 400, 500 and find 200 performs the best. 200 is used in all following experiments. We apply Equation 5 to quantize only self-attention and point-wise feed forward layers. Embedding layers are left unchanged due to the limit of time. But we believe that this does not affect the experiment results too much because previous work has found that 1 bit embeddings are as good as full precision ones [10].

4 Experiments

In this section, we show results of quantized transformer on machine translation, sentence classification, and question answering. All experiments are done on a single Nvidia Titan Xp GPU.

4.1 Machine Translation

We evaluate the performance of quantized transformer model on WMT 2014 English-to-German dataset, which consists of 4.5 million sentence pairs. We use the base transformer model provided by OpenNMT[1]. The base model's encoder and decoder each has 6 layers. Each layer has 8 attention heads, with 512 input dimensionality and 2048 inner-layer dimensionality. OpenNMT provides a pretrained full precision model with BLEU score 27.83. All the experiment hyperparameters are the same as in [1].

We first apply 1 bit weight only quantization to the model and compare the difference between training quantized model from scratch and initializing it from full precision one. The purpose of the experiment is to verify that training from scratch is much slower [6]. In Figure 2, we can see that the 1 bit model initialized from full precision one indeed trains much faster and achieves better accuracy on training set. For all the following experiments, we always initialize our model from full precision one.

Next, we train 1, 4, and 8 bits quantized models and evaluate their BLEU scores. From the training curve in Figure 3, we can see that three models converge to different training accuracy and at different speed. 8 bits model has the highest accuracy and converges the fastest. 4 bits model has the medium training accuracy and medium convergence speed. 1 bit model is the slowest to converge and performs the worst. Table 1 shows their corresponding BLEU scores. 8 bits model has the highest score among three quantized models and is only one point less than the 32 bits model. The original 32 bits model has size $\sim$ 500MB. After 8 bits quantization, the model size is reduced to $\sim$ 125MB. Given that an

[1] https://github.com/OpenNMT

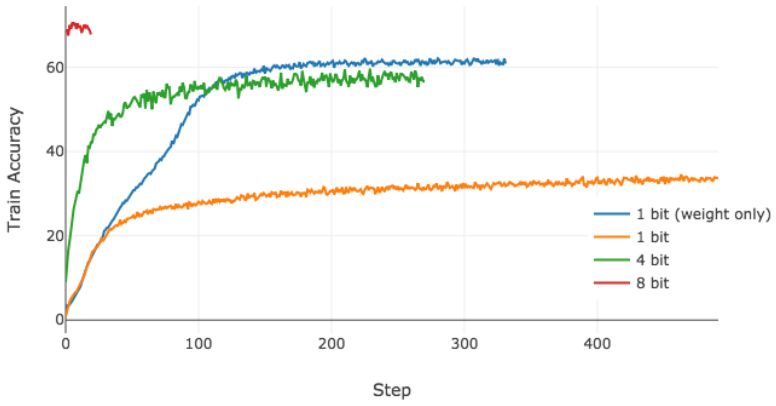

Figure 3: Training curves for 1, 4, 8 bit quantized transformer models.

Table 1: Quantized transformer's BLEU scores on WMT-14 EN-DE dataset

Model	BLEU
32-bit	**27.83**
8-bit	26.94
4-bit	23.18
1-bit	2.50
1-bit(weight only)	23.88

80 MB quantized RNN transducer speech recognition model can run smoothly on mobile devices [11], we believe that with some further optimizations, a transformer machine translation should also run on mobile devices, providing faster and offline translation experience.

1 bit quantized model perform very poorly. It has been observed that activations are more sensitive to quantization than weights [16]. Therefore we conduct another 1 bit weight only quantization experiment to check whether this is true in our setup. From Figure 3 and Table 1 we can see that 1 bit weight only quantized model performs even better than 4 bits model. This shows that transformer model's activations are indeed much more sensitive than weights to quantization. This result is also encouraging because it may be possible to train a transformer model that is wider and deeper so even when quantized it is still better and smaller than the origin 32 bits one.

4.2 BERT

BERT is the SOTA model for many NLP tasks [2]. Because we do not have the original data used to train BERT, we only experiment quantize BERT and fine-tuning it. As we have seen in previous translation experiments, 1 bit quantization requires much more training time than 4 and 8 bits quantization. Without access to lots of training data, we find that fine-tuning 1 bit quantized BERT model gets stuck on high training loss. So for BERT quantization, we only report results from 4 and 8 bits quantization.

For this experiment, we use this BERT implementation[2] with its pretrained BERT base model.

4.2.1 MRPC

MRPC (Microsoft Research Paraphrase Corpus) is a sentence classification tasks [17]. Its data consists of sentence pairs automatically extraceted from online news and annotated by human for

[2]https://github.com/huggingface/pytorch-pretrained-BERT

Table 2: Quantized BERT accuracy evaluation on MRPC dev set

Model	Accuracy
32-bit	84.6
8-bit	**86.3**
4-bit	68.4

Table 3: Quantized BERT exact match (EM) and F1 on SQuAD's dev set

Model	EM	F1
32-bit	**81.18**	**88.52**
8-bit	81.05	88.37

whether sentences in the pair are semantically equivalent. We use standard classification accuracy as evaluation metric.

We apply 4 and 8 bits quantization to the BERT base model and fine-tune it on MRPC dataset for 3 epochs with batch size 32, learning rate 2e-5, and max sequence length 128. Results are in Table 2. 8 bits quantized model performs slightly better than the full precision model and 4 bits quantized model perform much worse. The reason 8 bits model perform better may be that MRPC is a relatively small dataset so quantization works as regularization.

4.2.2 SQuAD

SQuAD (Stanford Question Answering Dataset) is a question answering task with 100k crowdsourced question/answer pairs [18]. We use SQuAD version 1.1 to make it comparable to the original BERT experiment. Standard exact match and F1 score are used as evaluation metric.

We only apply 8 bits quantization in this experiment. This is due to time limit and also because 4 bits quantization is likely to perform worse than 8 bits one. We fine-tune the model for 2 epochs, with batch size 12, learning rate 3e-5, maximum sequence length 384, and document stride 128.

Results are presented in Table 3. 8 bits model perform only slightly worse than 32 bit one.

5 Conclusion and Future Work

In this work, we experiment 1, 4, and 8 bits quantization on the original transformer model and the latest SOTA BERT model. We evaluate the quantized model's performance on machine translation, sentence classification, and question answering tasks. The results show that 8 bits quantization only hurts model's performance slightly. In cases where fine-tuning data is limited, model quantized to 8 bits might outperform the 32 bits one because quantization helps regularization. Given these results and recent success of quantized speech recognition model, we believe that it is possible to quantize transformer model and deploy it on mobile devices, enabling fast offline machine translation.

Note that this work is not a thorough study of quantization of transformer model. There are differences like symmetric and asymmetric quantization and more complicated quantization methods such as WRPN[19], DoReFa [16], and PACT[20] which we do not have time to explore. Also we do not run extensive experiments to select the best hyperparameters. Therefore it is possible to close gap between 8 bits and 32 bits models with further experiments. It is also possible for the quantized model to outperform the 32 bits one if quantized model is initialized with a bigger and deeper model. Besides quantization, other compression techniques like pruning and knowledge distillation may also help compress transformer model.

References

[1] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, pages 5998–6008, 2017.

[2] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[3] Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. Improving language understanding by generative pre-training. *URL https://s3-us-west-2. amazonaws. com/openai-assets/research-covers/languageunsupervised/language understanding paper. pdf*, 2018.

[4] Alex Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. 2019.

[5] High-performance hardware for machine learning. https://media.nips.cc/Conferences/2015/tutorialslides/Dally-NIPS-Tutorial-2015.pdf.

[6] Milad Alizadeh, Javier Fernández-Marqués, Nicholas D. Lane, and Yarin Gal. A systematic study of binary neural networks' optimisation. In *International Conference on Learning Representations*, 2019.

[7] Matthieu Courbariaux, Itay Hubara, Daniel Soudry, Ran El-Yaniv, and Yoshua Bengio. Binarized neural networks: Training deep neural networks with weights and activations constrained to+ 1 or-1. *arXiv preprint arXiv:1602.02830*, 2016.

[8] Philipp Gysel, Jon Pimentel, Mohammad Motamedi, and Soheil Ghiasi. Ristretto: A framework for empirical study of resource-efficient inference in convolutional neural networks. *IEEE transactions on neural networks and learning systems*, (99):1–6, 2018.

[9] 8-bit inference with tensorrt. http://on-demand.gputechconf.com/gtc/2017/presentation/s7310-8-bit-inference-with-tensorrt.pdf.

[10] Maximilian Lam. Word2bits-quantized word vectors. *arXiv preprint arXiv:1803.05651*, 2018.

[11] Yanzhang He, Tara N Sainath, Rohit Prabhavalkar, Ian McGraw, Raziel Alvarez, Ding Zhao, David Rybach, Anjuli Kannan, Yonghui Wu, Ruoming Pang, et al. Streaming end-to-end speech recognition for mobile devices. *arXiv preprint arXiv:1811.06621*, 2018.

[12] Mohammad Rastegari, Vicente Ordonez, Joseph Redmon, and Ali Farhadi. Xnor-net: Imagenet classification using binary convolutional neural networks. In *European Conference on Computer Vision*, pages 525–542. Springer, 2016.

[13] Song Han, Jeff Pool, John Tran, and William Dally. Learning both weights and connections for efficient neural network. In *Advances in neural information processing systems*, pages 1135–1143, 2015.

[14] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015.

[15] Neural networks for machine learning. Coursera, video lectures.

[16] Shuchang Zhou, Yuxin Wu, Zekun Ni, Xinyu Zhou, He Wen, and Yuheng Zou. Dorefa-net: Training low bitwidth convolutional neural networks with low bitwidth gradients. *arXiv preprint arXiv:1606.06160*, 2016.

[17] William B Dolan and Chris Brockett. Automatically constructing a corpus of sentential paraphrases. In *Proceedings of the Third International Workshop on Paraphrasing (IWP2005)*, 2005.

[18] Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. Squad: 100,000+ questions for machine comprehension of text. *arXiv preprint arXiv:1606.05250*, 2016.

[19] Asit Mishra, Eriko Nurvitadhi, Jeffrey J Cook, and Debbie Marr. Wrpn: wide reduced-precision networks. *arXiv preprint arXiv:1709.01134*, 2017.

[20] Jungwook Choi, Zhuo Wang, Swagath Venkataramani, Pierce I-Jen Chuang, Vijayalakshmi Srinivasan, and Kailash Gopalakrishnan. Pact: Parameterized clipping activation for quantized neural networks. *arXiv preprint arXiv:1805.06085*, 2018.