Fully Quantized Transformer for Improved Translation

Gabriele Prato
Mila, Université de Montréal
pratogab@mila.quebec

Ella Charlaix
Huawei Noah's Ark Lab
ella.charlaix@huawei.com

Mehdi Rezagholizadeh
Huawei Noah's Ark Lab
mehdi.rezagholizadeh@huawei.com

Abstract

State-of-the-art neural machine translation methods employ massive amounts of parameters. Drastically reducing computational costs of such methods without affecting performance has been up to this point unsolved. In this work, we propose a quantization strategy tailored to the Transformer [1] architecture. We evaluate our method on the WMT14 EN-FR and WMT14 EN-DE translation tasks and achieve state-of-the-art quantization results for the Transformer, obtaining no loss in BLEU scores compared to the unquantized baseline. We further compress the Transformer by showing that post-training, a good portion of the nodes in the encoder can be removed without causing any loss in BLEU. For the full version of this paper, please refer to: https://arxiv.org/abs/1910.10485

1 Introduction

The recent Transformer network [1] achieved state-of-the-art results on the WMT 2014 English-to-French and WMT 2014 English-to-German corpus. This self-attention based architecture inspired a new wave of work resulting in the state-of-the-art of numerous natural language processing tasks to reach new heights [2]. Unfortunately, these Transformer networks make use of an enormous amount of parameters. Inference on resource-limited hardware such as edge-devices is thus impractical.

A solution to reduce the computational burden of these neural networks is to lower numerical precision, which allows the representation of numerical values with fewer bits [3]. This method called quantization has the advantage of providing good compression rates with minimal accuracy loss and is supported by a great number of different hardware. Properly quantizing the Transformer would thus allow computational speed gains at inference, as well as deployment on more constrained hardware.

Recently, simple quantization solutions have been applied to the Transformer. Tierno (2019) [4] uses 8-bit fixed and linear quantization schemes on both the weights and inputs of Transformer layers. Cheong and Daniel (2019) [5] apply k-means quantization and binarization with two centroids over the weights of the Transformer network. Fan (2019) [6] uses binary and range based linear quantization on the Transformer. Bhandare et al. (2019) [7] quantize some of the MatMul operations of the Transformer and use the KL divergence to estimate the most suited parameters for each quantization range. So far, all proposed methods have failed to avoid any loss in translation quality and omit quantizing the whole Transformer architecture, resulting in suboptimal computational efficiency.

EMC^2: 5th Edition Co-located with NIPS'19

In this work, we propose a custom quantization strategy of the entire Transformer architecture, where quantization is applied throughout the whole training. Our method is easy to implement and results are consistent with the original Transformer. We test our approach on the WMT14 EN-FR and WMT14 EN-DE translation tasks and obtain state-of-the-art quantization results. We are, to the best of our knowledge, the first to fully quantize the Transformer architecture to 8-bit while maintaining the translation quality on par with the baseline and even achieve higher BLEU scores than the latter on some tasks.

2 Quantization Strategy

2.1 Quantization Method

We use the uniform quantization method described by [8]. Uniform quantization has the advantage of being easy to implement and is supported by most hardware.

Given an element x of a tensor $\mathbf{X}$, we apply the quantization function $\mathcal{Q}$:

$$\mathcal{Q}(x) = \frac{x - x_{min}}{s} \tag{1}$$

$$s = \frac{x_{max} - x_{min}}{2^k - 1} \tag{2}$$

where x_{min} and x_{max} are respectively $\min(\mathbf{X})$ and $\max(\mathbf{X})$ for weight quantization and running estimates for activation quantization. The latter are computed during training, where for every forward pass, the x_{min} and x_{max} variables are updated via an exponential moving average with a momentum of 0.9. In the context of 8-bit quantization, k is set to 8.

At training time, we simulate quantization by first quantizing and then rescaling to the original domain:

$$\left\lfloor \frac{\text{clamp}(x; x_{min}, x_{max}) - x_{min}}{s} \right\rceil * s + x_{min} \tag{3}$$

where the clamp function clamps all values outside the $[x_{min}, x_{max}]$ range and $\lfloor \cdot \rceil$ represents rounding to the nearest integer. During backpropagation, we use the straight-through estimator [9] and set the gradients of clamped values to zero. The only exception is for the LayerNorm's denominator, for which gradients are never zeroed, even though values can still be clamped. Once training is finished, s, x_{min} and x_{max} are frozen along with the weights.

2.2 What to Quantize

We choose to quantize all operations which will provide a computational speed gain at inference. In this regard, we quantize all matrix multiplications, meaning that the inputs and weights of MatMuls will both be 8-bit quantized. The other operations we quantize are divisions, but only if both the numerator and denominator are matrices or tensors, then we quantize them to 8-bit. For all other operations, such as sums, the computational cost added by the quantization operation outweighs the benefit of performing the operation with 8-bit inputs. Hence, we do not quantize such operations.

More precisely, we quantize all weights of the Transformer, excluding biases. The latter are summed with the INT32 output of matrix multiplications and thus provide no additional computational efficiency from being quantized. Furthermore, the memory space of biases is also insignificant in comparison to the weight matrices, representing less than 0.1% of total weights. For positional embeddings, memory gain is also minimal, but since these will be summed with the quantized input embeddings, we likewise quantize them. The γ weights of LayerNorms are also quantized. As for activations, we quantize the sum of the input embeddings with the positional encodings in both the encoder and decoder. In the Multi-Head Attention, we quantize the (Q, K, V) input, the softmax's numerator, the softmax's denominator, the softmax's output and the Scaled Dot-Product Attention's output. To be precise, in the Scaled Dot-Product Attention, we compute the softmax's denominator with the unquantized softmax's numerator and then quantize the numerator. For the position-wise feed-forward networks, we quantize the output of the ReLUs and of the feed-forward networks themselves. Finally, for all LayerNorms, we quantize the numerator $x - \mu$, the denominator $\sqrt{\sigma^2 + \epsilon}$, their quotient and the output of the LayerNorm.

Table 1: Our quantization strategy achieves better BLEU scores than all other quantization methods for the Transformer on the WMT14 EN-DE, WMT14 EN-FR and WMT17 EN-DE test set.

Model	Fully Quantized	BLEU		
		EN-DE (2014)	EN-FR	EN-DE (2017)
Vaswani et al. (2017) [1] (base)	No quantization	27.3	38.1	-
Cheong and Daniel (2019) [5]		-	-	27.38
Bhandare et al. (2019) [7]		27.33	-	-
Fan (2019) [6]		26.94	-	-
Our method (base)	✓	**27.60**	**39.12**	**27.60**

2.3 Bucketing

Instead of using a single set of (s, x_{min}, x_{max}) per quantized tensor, we can quantize subsets of a tensor using a set of (s, x_{min}, x_{max}) per subset [10]. Even though this adds more scalars, the memory cost is insignificant overall and the added flexibility can greatly alleviate the precision loss obtained by trying to fit all values of a tensor into a single domain with lower numerical precision.

We use this bucketing method for all weight matrices, where we bucket on the output dimension. That is, we have one set of (s, x_{min}, x_{max}) for every element of the output. For activations, we use bucketing when quantizing: the sum of input embeddings with the positional encoding, the Q, K, V inputs, the Scaled Dot-Product Attention's output, the feed-forward's output, the LayerNorm's numerator, the LayerNorm's quotient and the LayerNorm's output.

2.4 Dealing with Zeros

Unlike [8], we do not need to nudge the domain so that the zero value gets perfectly mapped. The only zero values which we have to deal with are the padding, the output of ReLU layers and dropouts. Since padding has no effect on the final output, we completely ignore these values when quantizing and when computing the running estimates x_{min} and x_{max}. For ReLUs, we fix the x_{min} estimate of those quantization layers to 0, which guarantees the perfect mapping of the value. Finally, quantization is applied before any dropout operation. Even though the zeros added to the output of the quantization layer might not be part of the domain, this only happens during training. At inference, no dropout is performed and thus quantization is not affected.

3 Experiments

3.1 Full Quantization

We apply our 8-bit quantization strategy on both the base and big Transformer [1]. The training setup of all presented models is the same as in the original paper, with the exception that the dropout ratio is set to 0.1 in all cases. We test our models on the WMT 2014 English-to-German and WMT 2014 English-to-French translation tasks. Reported perplexity is per token and BLEU was measured with `multi-bleu.pl`[1] on the `newstest2014`[2] test set. We used beam search with a beam size of 4 and a length penalty of 0.6, as in [1]. No checkpoint averaging was performed.

We compare our results with other 8-bit quantization methods in Table 1. Out of all the approaches, we are the only one fully quantizing the Transformer architecture.

In Table 2, we show performance of our method on the WMT14 EN-DE and WMT14 EN-FR after a fixed amount of training steps. We compare our results with our two non-quantized baseline Transformers (base and big variants). Training with quantization was about twice as slow as training the baselines. We also compare with two other quantization approaches. The first one is the "default" approach, which is to naively quantize every possible operation and the second approach applies our quantization strategy post-training (see section 3.3 for details). For post-training quantization, the

[1] `https://github.com/moses-smt/mosesdecoder/blob/master/scripts/generic/multi-bleu.perl`

[2] `https://www.statmt.org/wmt14/translation-task.html`

Table 2: Performance of our quantization method on the WMT14 EN-DE and WMT14 EN-FR test set for a fixed number of training steps.

Model	Quantized	Training Steps	Compression	EN-DE		EN-FR	
				PPL	BLEU	PPL	BLEU
Base		100k	1x	4.41	25.82	3.20	37.94
Default Approach	✓	100k	4x	74.04	0.21	*nan*	0
Post-Quantization	✓	100k	4x	4.45	25.50	3.22	37.96
Our method	✓	100k	4x	4.67	**26.98**	3.23	**38.55**
Big		300k	1x	4.03	26.85	2.72	40.17
Post-Quantization	✓	300k	4x	4.27	26.55	2.78	39.78
Our method	✓	300k	4x	4.24	**27.95**	2.80	**40.17**

Table 3: Effect of quantizing single activations of the Transformer on the translation quality. Results are on the WMT14 EN-FR test set.

Module	Quantized Activation	No Bucketing		Bucketing	
		PPL	BLEU	PPL	BLEU
Encoder	(Input Embedding + Positional Encoding)	3.20	38.61	3.20	39.08
Decoder	(Input Embedding + Positional Encoding)	3.20	39.35	3.20	39.36
Multi-Head Attention	Input (Q, K, V)	3.21	39.06	3.21	39.29
	LayerNorm Output	3.21	39.09	3.20	38.78
Scaled Dot-Product Attention	Softmax Numerator	3.20	39.32	3.21	39.01
	Softmax Denominator	3.21	39.35	3.21	39.11
	Softmax Output	3.22	39.41	3.22	38.87
	Output	3.21	38.73	3.21	39.02
Feed-forward	ReLU Output	3.21	39.43	3.22	38.93
	Feed-forward Output	3.54	38.03	3.20	39.27
	LayerNorm Output	3.21	38.67	3.21	39.04
LayerNorm	Numerator	3.53	37.75	3.21	38.86
	Denominator	1748	0	-	-
	Quotient	3.22	38.97	3.21	39.02

best BLEU score out of 20 trials was picked, with scores varying by about 0.2 BLEU. As for the default approach, the numerical stability provided by the ϵ in the LayerNorm's denominator is lost when quantizing every operation. This is why the default approach got *nan* in the EN-FR task.

3.2 Ablation Study

To compare the effect of bucketing and better understand which operation is more sensitive to quantization, we evaluate the effect of quantizing single operations of the Transformer, with and without bucketing. By single operation, we mean quantizing the operation of a module for all Transformer layers. Table 3 shows results on the WMT14 EN-FR translation task. The only operations underperforming our baseline are the LayerNorm's numerator when not bucketed and the denominator. The latter cannot be bucketed because all dimensions of the variance tensor vary per batch. Solely quantizing the LayerNorm's denominator with no bucketing works, but results are poor. To successfully quantize this element without causing performance issues, we suspect quantizing prior elements in the network helps, as is the case in our quantization scheme.

3.3 Delaying Quantization

Our method's goal is to increase computational efficiency when inferring with the Transformer. To this end, our quantization scheme only requires us to learn s and x_{min}. Although we do so with our

Table 4: Impact of delaying the learning of quantization parameters on translation quality. Results are on the WMT14 EN-DE and WMT14 EN-FR test set.

Quantization Start (training step)	EN-DE PPL	EN-DE BLEU	EN-FR PPL	EN-FR BLEU
100	4.67	**26.98**	3.23	38.55
10000	5.07	26.06	3.21	**38.62**
50000	5.04	26.48	3.21	38.50
80000	5.10	26.11	3.21	38.43
Post-Quantization	4.45	25.50	3.22	37.96

quantization scheme throughout the whole training, this is not a necessity. Quantization could be applied later on during training. Post-training quantization is also an option, where once the model is fully trained, we keep the weights fixed, but compute the s, x_{min} and x_{max} over a few hundred steps. We compare different starting points in Table 4, where we evaluate them on the WMT14 EN-DE and WMT14 EN-FR translation tasks.

Learning quantization parameters adds a significant computational cost during training. A major advantage to delaying quantization is to perform more training steps in the same given amount of time. Therefore, when training time is a constraint, one possible strategy is to train a model without quantization, to perform more training steps and then to post-quantize the model. Another advantage of post-quantization is that iterations can be quickly performed to search for the best performing candidate.

4 Conclusion

We proposed a quantization strategy for the Transformer, quantizing all operations which could provide a computational speed gain. With our method, we achieve higher BLEU scores than the baseline Transformer, as well as all other quantization methods on both WMT14 EN-DE and WMT14 EN-FR. We plan on applying our method to other tasks, as well as further exploring the compression of Transformers.

References

[1] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention Is All You Need," *arXiv e-prints*, p. arXiv:1706.03762, Jun 2017.

[2] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding," *arXiv e-prints*, p. arXiv:1810.04805, Oct 2018.

[3] M. Marchesi, G. Orlandi, F. Piazza, and A. Uncini, "Fast neural networks without multipliers," *IEEE Transactions on Neural Networks*, vol. 4, pp. 53–62, Jan 1993.

[4] A. Tierno, "Quantized Transformer," tech. rep., Stanford University, Stanford, California, 2019.

[5] R. Cheong and R. Daniel, "transformers.zip: Compressing Transformers with Pruning and Quantization," tech. rep., Stanford University, Stanford, California, 2019.

[6] C. Fan, "Quantized Transformer," 2019.

[7] A. Bhandare, V. Sripathi, D. Karkada, V. Menon, S. Choi, K. Datta, and V. Saletore, "Efficient 8-Bit Quantization of Transformer Neural Machine Language Translation Model," *arXiv e-prints*, p. arXiv:1906.00532, Jun 2019.

[8] 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," *arXiv e-prints*, p. arXiv:1712.05877, Dec 2017.

[9] G. Hinton, "Neural networks for machine learning," 2012. Coursera, video lectures.

[10] D. Alistarh, D. Grubic, J. Li, R. Tomioka, and M. Vojnovic, "QSGD: Communication-Efficient SGD via Gradient Quantization and Encoding," *arXiv e-prints*, p. arXiv:1610.02132, Oct 2016.