# FULLY QUANTIZED TRANSFORMER FOR IMPROVED TRANSLATION

**Anonymous authors**
Paper under double-blind review

## 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 (Vaswani et al., 2017) 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 non-quantized baseline. We further compress the Transformer by showing that, once the model is trained, a good portion of the nodes in the encoder can be removed without causing any loss in BLEU.

## 1 INTRODUCTION

Neural machine translation methods have achieved impressive results lately (Ahmed et al., 2017; Ott et al., 2018; Edunov et al., 2018). Having been proposed only recently (Kalchbrenner & Blunsom, 2013; Sutskever et al., 2014; Cho et al., 2014), many great work have led the field to move forward quickly. Bahdanau et al. (2014) introduced an attention mechanism, allowing the decoder to attend to any hidden state generated by the encoder. Multiple improvements to their approach have been proposed, such as multiplicative attention (Luong et al., 2015) and more recently multi-head self-attention (Vaswani et al., 2017). The latter's novel Transformer architecture achieved state-of-the-art results on the WMT 2014 English-to-French and WMT 2014 English-to-German corpus. Inspiring a new wave of work, state-of-the-art of numerous natural language processing tasks have reached new heights (Devlin et al., 2018; Liu et al., 2019). Unfortunately, these Transformer networks make use of an enormous amount of parameters, resulting in impractical inference on more limited hardware such as edge-devices is 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 (Tang & Kwan, 1993; Marchesi et al., 1993). 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 limited hardware.

Recently, simple quantization solutions have been applied to the Transformer. Tierno (2019) uses 8-bit fixed and linear quantization schemes on both the weights and inputs of Transformer layers. Cheong & Daniel (2019) apply $k$-means quantization and binarization with two centroids over the weights of the Transformer network. Fan (2019) uses binary and range based linear quantization on the Transformer. Bhandare et al. (2019) 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.

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 non-quantized Transformer. We test our approach on the WMT 14 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 non-quantized baseline and even achieve

higher BLEU scores than the latter on some tasks. We also show that, once models are trained, a great number of nodes in the encoder can be removed without affecting translation quality. Our pruning scheme is adaptive, meaning it estimates the right ratio of nodes to prune per layer. It also requires no extra training and provides additional compression when combined with quantization.

## 2 Related Work

In this section, we review a broad spectrum of quantization and pruning methods for neural network compression.

### 2.1 Quantization

Over the years, a broad range of methods have been proposed to quantize neural networks. These include, among many others, binary (Courbariaux et al., 2016), ternary (Lin et al., 2015; Li et al., 2016), uniform affine (Jacob et al., 2017) and learned (Zhang et al., 2018) quantization.

Quantization has been applied to RNNs (Jordan, 1990), LSTMs (Hochreiter & Schmidhuber, 1997) and GRUs (Cho et al., 2014). Ott et al. (2016) propose an exponential quantization method for RNN weights. They compare their method with binary and ternary quantization on language modelling and speech recognition and show binary weights to be ineffective for RNNs, while ternary and exponential quantization to work well. Hubara et al. (2016) quantize weights and activations of RNNs and LSTMs to 2, 4 and 6 bits, while keeping gradients in full precision. He et al. (2016) propose modifications to the gates and interlinks of quantized LSTM and GRU cells, as well as a balanced quantization method for weights. Wu et al. (2016) successfully quantize the GNMT neural machine translation model to 8-bit without any loss in translation quality. Wang et al. (2018) propose to use different quantization methods for different RNN components.

With regards to CNNs (LeCun et al., 1989), various works have explored quantizing the architecture. Gong et al. (2014) compare matrix factorization, binarization, $k$-means clustering, product quantization and residual quantization on CNNs. Wu et al. (2015) apply quantization to both kernels and fully connected layers of convolutional neural networks. Rastegari et al. (2016) evaluate the use of binary weights on AlexNet (Krizhevsky et al., 2012). Zhou et al. (2016) use low bitwidth weights, activations and gradients on CNNs.

Quantization has also been jointly used with other compression methods. Han et al. (2015) combine pruning, quantization, weight sharing and Huffman coding, while Polino et al. (2018) use quantization with knowledge distillation (Hinton et al., 2015) for superior compression rates.

### 2.2 Pruning

LeCun et al. (1990) were the first to propose a Hessian based method to prune neural net weights, with Hassibi et al. (1994) later improving the method. More recently, See et al. (2016) show that pruning a fully trained model and then retraining it can increase performance over the original non-pruned model. Gradually pruning in tandem with training has also been shown to increase performance (Zhu & Gupta, 2017). Liu et al. (2017) prune nodes instead of weights by applying a penalty in the loss on the $\gamma$ parameters of batch normalization layers. Narang et al. (2017b) make better use of hardware by applying pruning and weight decay in blocks to minimize the number of loaded weight matrix chunks. Chen et al. (2018) combine quantization with block based low-rank matrix approximation of embeddings.

For pruning methods applied to specific architectures, Liu et al. (2015) propose an efficient sparse matrix multiplication algorithm for CNNs. For RNNs, Narang et al. (2017a) show sparse pruning to work on the architecture. In order to maintain dimension consistency, Wen et al. (2017) propose to prune all basic LSTM structures concurrently. Park et al. (2018) introduce simple recurrent units (SRUs) for easy pruning of RNNs.

## 3 Quantization Strategy

In this section, we describe our custom quantization scheme for the Transformer.

### 3.1 QUANTIZATION METHOD

We use the uniform quantization method described by Jacob et al. (2017). Uniform quantization has the advantage of being easy to implement and is supported by most hardware.

At inference, 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 running estimates are computed during training, where for every forward pass, the variables $x_{min}$ and $x_{max}$ are updated with 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 $\text{clamp}$ function clamps all values outside the $[x_{min}, x_{max}]$ range and $\lfloor \cdot \rceil$ the rounding to the nearest integer operator. During backpropagation, we use the straight-through estimator (Hinton, 2012) 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.

### 3.2 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 operation 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 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 $\gamma$ 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.

### 3.3 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 (Alistarh et al., 2016). 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 single 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

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) | BLEU EN-FR | BLEU EN-DE (2017) |
|---|---|---|---|---|
| Vaswani et al. (2017) (base) | No quantization | 27.3 | 38.1 | - |
| Cheong & Daniel (2019) | | - | - | 27.38 |
| Bhandare et al. (2019) | | 27.33 | - | - |
| Fan (2019) | | 26.94 | - | - |
| Our method (base) | ✓ | **27.60** | **39.12** | **27.60** |

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.

### 3.4 Dealing with Zeros

Unlike Jacob et al. (2017), 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 time. At inference, no dropout is performed and thus quantization is not affected.

## 4 Experiments

In this section, we present the results of our full quantization scheme as well as the effect of applying quantization to individual Transformer operations. We also compare translation quality when starting to apply our quantization method at different moments during training. Finally, we show the result of node pruning on the trained models.

### 4.1 Full Quantization

We apply our 8-bit quantization strategy on both the base and big Transformer (Vaswani et al., 2017) and follow the same training setup as in the original paper, except for the dropout ratio, which we 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 (Vaswani et al., 2017). 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). The baselines are trained with the same setup as the quantized variants, without quantization, which is the same setup as Vaswani et al. (2017). 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

[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** |

section 4.3 for details). For post-training quantization, the best BLEU score out of 20 trials was picked, with scores varying by about 0.2 BLEU. For the big Transformer variants, we omitted to bucket the Scaled Dot-Product Attention's output and the sum of the decoder's input embeddings with the positional encoding. Bucketing these activations might provide better results than what we have obtained.

In 3 out of 4 cases, our quantization strategy scores the highest BLEU even though the baselines benefit from full precision. In the case of EN-FR, our method scored equal BLEU with the non-quantized big variant. Generally, fully quantizing the Transformer seems to result in a loss in perplexity while usually increasing translation accuracy. The reason for this might be the lower numerical precision acting as a regularization effect. Indeed, when quantizing the big Transformer on the WMT14 EN-DE task, a training corpus which the model can easily overfit, we saw a net gain of 1.1 BLEU for the same amount of training.

As for the two other quantization approaches, post-quantization slightly increased the BLEU on the WMT14 EN-FR for the base Transformer, but in every other case the translation quality diminished. For the default approach, the method performed very poorly. By quantizing every operation, we lose the numerical stability provided by the $\epsilon$ in the LayerNorm's denominator. This is why the default approach got *nan* in the EN-FR task.

### 4.2 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 non-quantized baseline of 37.94 BLEU 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.

We also tested quantizing solely the softmax output in the attention for a big Transformer variant on the WMT14 EN-DE task and saw an increase of 1.32 BLEU over the non quantized baseline at 300k training steps. For the same setup, but on WMT14 EN-FR, we did not notice any gain in BLEU.

### 4.3 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 quantization scheme throughout the whole training, it is not a necessity. Quantization could also be applied later on while 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

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 |

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 | | EN-FR | |
|---|---|---|---|---|
| | PPL | BLEU | PPL | BLEU |
| Base (no quantization) | 4.41 | 25.85 | 3.20 | 37.94 |
| 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 |

steps. We compare different starting points in Table 4, where we evaluate them on the WMT14 EN-DE and WMT14 EN-FR translation tasks. From our observed results, quantizing the model early on seems preferable, further reinforcing our belief that quantization helps training via regularization.

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.

### 4.4 Pruning Useless Nodes

Once the model is fully trained and quantized, we can further compress it by removing useless nodes. By useless, we mean nodes which do not cause any loss in translation quality when removed. We choose to prune nodes instead of independently pruning weights, to avoid the need of any special hardware or software, which is usually the case when trying to leverage sparse weight matrices obtained by the latter method. Pruning nodes results in concretely shrunken models. When getting rid of a node, we remove its corresponding set of weights from the layer outputting it and the following layer receiving the node as input.

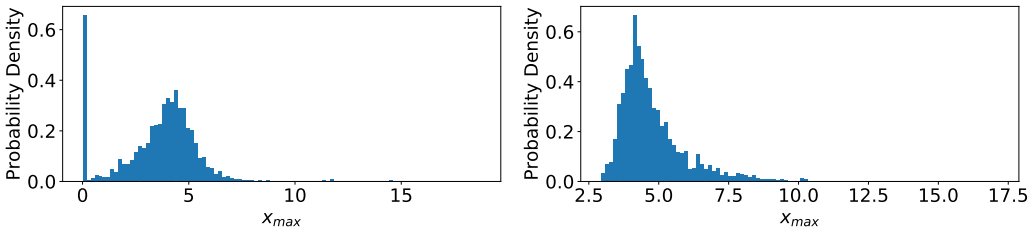


Figure 1: Histograms of the running estimates $x_{max}$ of a ReLU layer in the encoder (left) and decoder (right), one $x_{max}$ per output node of the layer.

The only nodes of the Transformer which can be removed without causing alterations to other components of the network are the nodes in between the two layers of each feed-forward networks. Fortunately, these consist of a substantial portion of the model's weights. In the case of the base Transformer, for a respective vocabulary of size 37000 and 32000, 39.96% and 41.65% of the total weights are owned by the feed-foward networks. This number grows to 47.03% and 48.18% in the big Transformer, for again, a respective vocabulary of size 37000 and 32000.

To evaluate which nodes can be safely pruned without affecting translation quality, we estimate the maximum value $x_{max}$ for each node of the ReLU output over a few hundred steps. This is done on the training set, using the fully trained model and keeping all other weights frozen. The $x_{max}$ are computed before quantizing the ReLU output and do not replace the ones used by the quantization process. Figure 1 shows the histogram of these running estimates for one ReLU layer in the encoder and one in the decoder. All other ReLU layers share the same pattern, where in the encoder there are always multiple $x_{max}$ close to 0, while it is not the case for the decoder.

Once the running estimates are computed, we prune its corresponding node if $x_{max} < z\sigma$ where $z$ is a hyperparameter and $\sigma$ the standard deviation of the $x_{max}$ of the layer. We empirically found $z = 0.025$ to work best, with higher thresholds causing BLEU to quickly decay. No retraining of the model is performed after nodes have been pruned.

Using this pruning method, we can compress and even slightly increase the BLEU scores of the non-pruned models. Table 5 shows results of our pruning method. Our approach has the advantage of being adaptive, meaning the number of nodes pruned per layer will differ as opposed to a fixed pruning ratio method. For example, in the case of the big Transformer trained on WMT14 EN-FR, 169 nodes were pruned in the first ReLU of the encoder, while in the second, 1226 were pruned. Nodes in the decoder rarely got pruned, at most 4 in the whole decoder.

The threshold allows us to find the right ratio of nodes to prune per layer instead of the usual: decide the ratio first and then prune. We compared with two such methods, where for each task, we fix the ratio to the global percentage of nodes pruned in the encoder by our method. The first fixed pruning method uses L1 norm to sort nodes to prune in ascending order, while the second sorts nodes using the $x_{max}$, also in ascending order.

With $x_{max}$ being running estimates, results varied per trial. Though with the method only taking a few hundred training steps to perform, running many trials is not an issue. We had to run at most four trials per task to improve BLEU score. When not improving, BLEU would either be the same or decreased by about $0.01-0.02$.

## 5 Conclusion

We proposed a quantization strategy for the Transformer, quantizing all operations which could provide a computational speed gain for a fully quantized architecture. We also proposed pruning unimportant nodes in the Transformer's encoder for further compression of the network. All of our design decisions were aimed at maximizing computational efficiency while making sure our method would benefit as many different types of hardware as possible.

Table 5: Comparison of our adaptive pruning scheme versus fixed rate pruning methods for equal pruning proportions. Total compression is of quantization combined with pruning.

| Pruning Method | Task | PPL | BLEU | Nodes Pruned in Encoder FF | Total Compression |
|---|---|---|---|---|---|
| No pruning (base) | EN-DE | 4.39 | 27.60 | 0% | 4x |
| Fixed pruning, L1 norm sort | EN-DE | 5.57 | 23.99 | 13.57% | 4.12x |
| Fixed pruning, $x_{max}$ sort | EN-DE | 4.57 | 27.33 | 13.57% | 4.12x |
| Adaptive pruning, $x_{max}$ threshold | EN-DE | 4.40 | **27.61** | 13.57% | 4.12x |
| No pruning (big) | EN-DE | 4.24 | 27.95 | 0% | 4x |
| Fixed pruning, L1 norm sort | EN-DE | 5.80 | 22.65 | 26.39% | 4.27x |
| Fixed pruning, $x_{max}$ sort | EN-DE | 4.47 | 27.43 | 26.39% | 4.27x |
| Adaptive pruning, $x_{max}$ threshold | EN-DE | 4.25 | **27.97** | 26.39% | 4.27x |
| No pruning (base) | EN-FR | 2.90 | 39.12 | 0% | 4x |
| Fixed pruning, L1 norm sort | EN-FR | 4.25 | 25.97 | 10.92% | 4.09x |
| Fixed pruning, $x_{max}$ sort | EN-FR | 3.10 | 38.42 | 10.92% | 4.09x |
| Adaptive pruning, $x_{max}$ threshold | EN-FR | 2.90 | **39.12** | 10.92% | 4.09x |
| No pruning (big) | EN-FR | 2.80 | 40.17 | 0% | 4x |
| Fixed pruning, L1 norm sort | EN-FR | 4.16 | 28.85 | 28.41% | 4.29x |
| Fixed pruning, $x_{max}$ sort | EN-FR | 2.91 | 39.40 | 28.41% | 4.29x |
| Adaptive pruning, $x_{max}$ threshold | EN-FR | 2.80 | **40.18** | 28.41% | 4.29x |

With our method, we achieve higher BLEU scores than all other quantization methods for the Transformer on both the WMT14 EN-DE and WMT14 EN-FR translation tasks and avoid any loss in BLEU compared to our non-quantized baseline Transformer, even obtaining higher BLEU than the latter. Our work shows that it is possible to fully quantize the Transformer to 8-bit and outperform the full precision model.

Previous work on Transformer quantization are unspecific in the details of their implementation. This makes it difficult for us to compare differences between methods. In our case, we aimed at minimizing the loss of information when quantizing every single operation. We found bucketing, careful handling of zeros and not clamping gradients of the quantized LayerNorm's denominator crucial for our method to work.

We are very excited about the possibilities this opens and plan on applying our method to other tasks. We also intend to extend our work to variations of the Transformer as well as further exploring the compression of these networks.

## REFERENCES

Karim Ahmed, Nitish Shirish Keskar, and Richard Socher. Weighted Transformer Network for Machine Translation. *arXiv e-prints*, art. arXiv:1711.02132, Nov 2017.

Dan Alistarh, Demjan Grubic, Jerry Li, Ryota Tomioka, and Milan Vojnovic. QSGD: Communication-Efficient SGD via Gradient Quantization and Encoding. *arXiv e-prints*, art. arXiv:1610.02132, Oct 2016.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural Machine Translation by Jointly Learning to Align and Translate. *arXiv e-prints*, art. arXiv:1409.0473, Sep 2014.

Aishwarya Bhandare, Vamsi Sripathi, Deepthi Karkada, Vivek Menon, Sun Choi, Kushal Datta, and Vikram Saletore. Efficient 8-Bit Quantization of Transformer Neural Machine Language Translation Model. *arXiv e-prints*, art. arXiv:1906.00532, Jun 2019.

Patrick Chen, Si Si, Yang Li, Ciprian Chelba, and Cho-Jui Hsieh. Groupreduce: Block-wise low-rank approximation for neural language model shrinking. In S. Bengio, H. Wallach, H. Larochelle,

K. Grauman, N. Cesa-Bianchi, and R. Garnett (eds.), *Advances in Neural Information Processing Systems 31*, pp. 10988–10998. Curran Associates, Inc., 2018.

Robin Cheong and Robel Daniel. transformers.zip: Compressing Transformers with Pruning and Quantization. Technical report, Stanford University, Stanford, California, 2019. URL `https://web.stanford.edu/class/cs224n/reports/custom/15763707.pdf`.

Kyunghyun Cho, Bart van Merrienboer, Dzmitry Bahdanau, and Yoshua Bengio. On the properties of neural machine translation: Encoder–decoder approaches. In *Proceedings of SSST-8, Eighth Workshop on Syntax, Semantics and Structure in Statistical Translation*, pp. 103–111, Doha, Qatar, October 2014. Association for Computational Linguistics. doi: 10.3115/v1/W14-4012.

Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation. *arXiv e-prints*, art. arXiv:1406.1078, Jun 2014.

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 e-prints*, art. arXiv:1602.02830, Feb 2016.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. *arXiv e-prints*, art. arXiv:1810.04805, Oct 2018.

Sergey Edunov, Myle Ott, Michael Auli, and David Grangier. Understanding Back-Translation at Scale. *arXiv e-prints*, art. arXiv:1808.09381, Aug 2018.

Chaofei Fan. Quantized Transformer. Technical report, Stanford University, Stanford, California, 2019. URL `https://web.stanford.edu/class/cs224n/reports/custom/15742249.pdf`.

Yunchao Gong, Liu Liu, Ming Yang, and Lubomir Bourdev. Compressing Deep Convolutional Networks using Vector Quantization. *arXiv e-prints*, art. arXiv:1412.6115, Dec 2014.

Song Han, Huizi Mao, and William J. Dally. Deep Compression: Compressing Deep Neural Networks with Pruning, Trained Quantization and Huffman Coding. *arXiv e-prints*, art. arXiv:1510.00149, Oct 2015.

Babak Hassibi, David G. Stork, and Gregory Wolff. Optimal brain surgeon: Extensions and performance comparisons. In J. D. Cowan, G. Tesauro, and J. Alspector (eds.), *Advances in Neural Information Processing Systems 6*, pp. 263–270. Morgan-Kaufmann, 1994.

Qinyao He, He Wen, Shuchang Zhou, Yuxin Wu, Cong Yao, Xinyu Zhou, and Yuheng Zou. Effective Quantization Methods for Recurrent Neural Networks. *arXiv e-prints*, art. arXiv:1611.10176, Nov 2016.

Geoffrey Hinton. Neural networks for machine learning, 2012. Coursera, video lectures.

Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the Knowledge in a Neural Network. *arXiv e-prints*, art. arXiv:1503.02531, Mar 2015.

Sepp Hochreiter and Jrgen Schmidhuber. Long short-term memory. *Neural computation*, 9:1735–80, 12 1997. doi: 10.1162/neco.1997.9.8.1735.

Itay Hubara, Matthieu Courbariaux, Daniel Soudry, Ran El-Yaniv, and Yoshua Bengio. Quantized Neural Networks: Training Neural Networks with Low Precision Weights and Activations. *arXiv e-prints*, art. arXiv:1609.07061, Sep 2016.

Benoit Jacob, Skirmantas Kligys, Bo Chen, Menglong Zhu, Matthew Tang, Andrew Howard, Hartwig Adam, and Dmitry Kalenichenko. Quantization and Training of Neural Networks for Efficient Integer-Arithmetic-Only Inference. *arXiv e-prints*, art. arXiv:1712.05877, Dec 2017.

Michael I. Jordan. Artificial neural networks. chapter Attractor Dynamics and Parallelism in a Connectionist Sequential Machine, pp. 112–127. IEEE Press, Piscataway, NJ, USA, 1990. ISBN 0-8186-2015-3.

Nal Kalchbrenner and Phil Blunsom. Recurrent continuous translation models. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pp. 1700–1709, Seattle, Washington, USA, October 2013. Association for Computational Linguistics.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger (eds.), *Advances in Neural Information Processing Systems 25*, pp. 1097–1105. Curran Associates, Inc., 2012.

Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel. Backpropagation applied to handwritten zip code recognition. *Neural Computation*, 1(4):541–551, Dec 1989. ISSN 0899-7667. doi: 10.1162/neco.1989.1.4.541.

Yann LeCun, John S. Denker, and Sara A. Solla. Optimal brain damage. In D. S. Touretzky (ed.), *Advances in Neural Information Processing Systems 2*, pp. 598–605. Morgan-Kaufmann, 1990.

Fengfu Li, Bo Zhang, and Bin Liu. Ternary Weight Networks. *arXiv e-prints*, art. arXiv:1605.04711, May 2016.

Zhouhan Lin, Matthieu Courbariaux, Roland Memisevic, and Yoshua Bengio. Neural Networks with Few Multiplications. *arXiv e-prints*, art. arXiv:1510.03009, Oct 2015.

Baoyuan Liu, Min Wang, Hassan Foroosh, Marshall Tappen, and Marianna Pensky. Sparse convolutional neural networks. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2015.

Xiaodong Liu, Pengcheng He, Weizhu Chen, and Jianfeng Gao. Multi-Task Deep Neural Networks for Natural Language Understanding. *arXiv e-prints*, art. arXiv:1901.11504, Jan 2019.

Zhuang Liu, Jianguo Li, Zhiqiang Shen, Gao Huang, Shoumeng Yan, and Changshui Zhang. Learning Efficient Convolutional Networks through Network Slimming. *arXiv e-prints*, art. arXiv:1708.06519, Aug 2017.

Minh-Thang Luong, Hieu Pham, and Christopher D. Manning. Effective Approaches to Attention-based Neural Machine Translation. *arXiv e-prints*, art. arXiv:1508.04025, Aug 2015.

M. Marchesi, G. Orlandi, F. Piazza, and A. Uncini. Fast neural networks without multipliers. *IEEE Transactions on Neural Networks*, 4(1):53–62, Jan 1993. ISSN 1045-9227. doi: 10.1109/72.182695.

Sharan Narang, Erich Elsen, Gregory Diamos, and Shubho Sengupta. Exploring Sparsity in Recurrent Neural Networks. *arXiv e-prints*, art. arXiv:1704.05119, Apr 2017a.

Sharan Narang, Eric Undersander, and Gregory Diamos. Block-Sparse Recurrent Neural Networks. *arXiv e-prints*, art. arXiv:1711.02782, Nov 2017b.

Joachim Ott, Zhouhan Lin, Ying Zhang, Shih-Chii Liu, and Yoshua Bengio. Recurrent Neural Networks With Limited Numerical Precision. *arXiv e-prints*, art. arXiv:1608.06902, Aug 2016.

Myle Ott, Sergey Edunov, David Grangier, and Michael Auli. Scaling Neural Machine Translation. *arXiv e-prints*, art. arXiv:1806.00187, Jun 2018.

Jinhwan Park, Yoonho Boo, Iksoo Choi, Sungho Shin, and Wonyong Sung. Fully neural network based speech recognition on mobile and embedded devices. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett (eds.), *Advances in Neural Information Processing Systems 31*, pp. 10620–10630. Curran Associates, Inc., 2018.

Antonio Polino, Razvan Pascanu, and Dan Alistarh. Model compression via distillation and quantization. *arXiv e-prints*, art. arXiv:1802.05668, Feb 2018.

Mohammad Rastegari, Vicente Ordonez, Joseph Redmon, and Ali Farhadi. XNOR-Net: ImageNet Classification Using Binary Convolutional Neural Networks. *arXiv e-prints*, art. arXiv:1603.05279, Mar 2016.

Abigail See, Minh-Thang Luong, and Christopher D. Manning. Compression of Neural Machine Translation Models via Pruning. *arXiv e-prints*, art. arXiv:1606.09274, Jun 2016.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. Sequence to sequence learning with neural networks. In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger (eds.), *Advances in Neural Information Processing Systems 27*, pp. 3104–3112. Curran Associates, Inc., 2014.

C. Z. Tang and H. K. Kwan. Multilayer feedforward neural networks with single powers-of-two weights. *IEEE Transactions on Signal Processing*, 41(8):2724–2727, Aug 1993. ISSN 1053-587X. doi: 10.1109/78.229903.

Andrew Tierno. Quantized Transformer. Technical report, Stanford University, Stanford, California, 2019. URL `https://web.stanford.edu/class/cs224n/reports/custom/15848474.pdf`.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention Is All You Need. *arXiv e-prints*, art. arXiv:1706.03762, Jun 2017.

Peiqi Wang, Xinfeng Xie, Lei Deng, Guoqi Li, Dongsheng Wang, and Yuan Xie. Hitnet: Hybrid ternary recurrent neural network. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett (eds.), *Advances in Neural Information Processing Systems 31*, pp. 604–614. Curran Associates, Inc., 2018.

Wei Wen, Yuxiong He, Samyam Rajbhandari, Minjia Zhang, Wenhan Wang, Fang Liu, Bin Hu, Yiran Chen, and Hai Li. Learning Intrinsic Sparse Structures within Long Short-Term Memory. *arXiv e-prints*, art. arXiv:1709.05027, Sep 2017.

Jiaxiang Wu, Cong Leng, Yuhang Wang, Qinghao Hu, and Jian Cheng. Quantized Convolutional Neural Networks for Mobile Devices. *arXiv e-prints*, art. arXiv:1512.06473, Dec 2015.

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V. Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, Jeff Klingner, Apurva Shah, Melvin Johnson, Xiaobing Liu, Łukasz Kaiser, Stephan Gouws, Yoshikiyo Kato, Taku Kudo, Hideto Kazawa, Keith Stevens, George Kurian, Nishant Patil, Wei Wang, Cliff Young, Jason Smith, Jason Riesa, Alex Rudnick, Oriol Vinyals, Greg Corrado, Macduff Hughes, and Jeffrey Dean. Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation. *arXiv e-prints*, art. arXiv:1609.08144, Sep 2016.

Dongqing Zhang, Jiaolong Yang, Dongqiangzi Ye, and Gang Hua. LQ-Nets: Learned Quantization for Highly Accurate and Compact Deep Neural Networks. *arXiv e-prints*, art. arXiv:1807.10029, Jul 2018.

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 e-prints*, art. arXiv:1606.06160, Jun 2016.

Michael Zhu and Suyog Gupta. To prune, or not to prune: exploring the efficacy of pruning for model compression. *arXiv e-prints*, art. arXiv:1710.01878, Oct 2017.