Quantized Transformer

Andrew Tierno
Department of Computer Science
Stanford University
Stanford, CA
atierno@stanford.edu

Abstract

1 Introduction

The Transformer has become an increasingly popular choice of neural architecture for language based tasks. Perhaps most famously, Devlin et al.'s BERT contextual word embeddings have yielded state of the art results on a range of language tasks, and are constructed with a network that uses Transformers as a basic unit. [Devlin et al., 2018] Given this rise in popularity, we aim to see if we can replicate the success of the Transformer while reducing the required space to store the model and the required time to train it. In this paper, we discuss the results of applying the quantization techniques proposed by Hubara et al. in their work "Quantized Neural Networks: Training Neural Networks with Low Precision Weights and Activations" to the Transformer in order to reduce the bitwidth required for the weights and take advantage of binary operations to speed up computation time. [Itay Hubara, 2018] For the purposes of this investigation, we focused on the techniques for quantizing weights and activations as opposed to the suggested shift based batch normalization and shift based AdaMax, as even quantizing the weights could yield a 4x reduction in size of the network. We have found that the Transformer has struggled to adapt to the quantized scheme, and has so far failed to meaningfully learn on the IWSLT 2014 Vietnamese-English translation task. We shall analyze potential explanations for why and future adaptations to experiment with that may yield more promising results.

2 Related Work

Our work is primarily inspired by the work of Hubara et al.'s "Quantized Neural Networks: Training Neural Networks with Low Precision Weights and Activations", which suggests a general framework for learning quantized representations of feed forward neural networks. Namely, this technique involves applying a quantization scheme to the parameters of a particular layer while still storing the weights internally in their full floating point precision until evaluation time. This paper builds off their previous work in which they develop fully binarized networks, i.e. those using only +1 and -1 as weights. [Courbariaux and Bengio, 2016]

As a technique, Quantized Neural Networks (QNNs) have been used to some extent in the image space. One notable example is XNOR-Net a binary neural network for image classification on ImageNet. [Rastegari et al., 2016] Beyond low bit network techniques, other researchers are looking into various ways to make networks smaller and more efficient. Some examples include SqueezeNet and MobileNet. [Iandola et al., 2016, Howard et al., 2017]

There appear to be few papers that explore the applications of low bit networks in a language space. One notable example would be Word2Bits, which works on training Word2vec vectors

Preprint. Work in progress.

in a low bit space. [Lam, 2018] Further, Hubara et al.'s paper builds out an LSTM and evaluates it over the Penn Treebank dataset, finding near comparable results to the original network while saving on space, and remark that language would be a viable next step for these low bit networks. [Itay Hubara, 2018]

To the best of the knowledge of the authors of this paper, there are no papers currently published that attempt to quantize the Transformer model or use low bit networks for a translation task.

3 Approach

3.1 Quantized Neural Networks

The key insight that drives QNNs is that during training time, all weights are stored as real valued numbers, yet the activation of a particular layer and the weight in the following layer are quantized before computing the new activation. Quantization of a particular number, they note, can be expressed through

$$\text{LinearQuant}(x, bitwidth) = \text{Clip}\left(\text{round}\left(\frac{x}{bitwidth})\right) \times bitwidth, minVal, maxVal\right)$$

where $bitwidth$ is the desired size in bits of each weight or activation, $minVal$ and $maxVal$ respectively are set to some constants to bound the quantization. Practically, these constants are set to be 0 for $minVal$ and the standard deviation of the values in the inputted tensor for the $maxVal$ times the bitwidth. The procedure for forward propagation then becomes an alternating pattern of quantizing the previous activation and current weights, computing their product, and passing this product into a batch normalization layer, whose output becomes the activation for the following layer. Passing a gradient backwards through a quantized layer has already been thoroughly studied, and notably Hinton and Bengio recommend using a "straight through estimator" (STE). [Bengio et al., 2013] Simply, the layer of quantization in the forward passed is replaced with an identity operation during backpropagation. The STE overcomes the primary issue with quantized systems that their gradient is almost everywhere zero. Courbairaux et. al propose a modification to the STE. When computing the gradient of a value r whose quantized form is $q = \text{Quant}(r, bitwidth)$ for some quantization function Quant, if the gradient with respect to q is g_q (as obtained with the vanilla STE), then

$$g_r = g_q 1_{minVal \leq |r| \leq maxVal}$$

for the specified constants $minVal, maxVal$.

We also experimented with an alternate scheme for quantization, namely fixed point quantization. For this scheme, a number is represented with $bitwidth$ bits, the last $precision$ of which represent the decimal part of the binary representation of the number. A number can be quantized under this scheme using

$$\text{FixedQuant}(x, precision, bitwidth) = \text{Clip}(\lfloor x \times 2^{precision} \rfloor \times 2^{-precision}, minVal, maxVal)$$

where $maxVal = \sum_{i=-precision}^{bitwidth-precision-1} 2^i$ and $minVal = -maxVal$. This scheme is notably quite similar to that of linear quantization, but relies instead on a fixed scheme rather than the distribution of the data.

3.2 Quantizing the Feed Forward Layers

For the feed forward layers we use the approach directly suggested by Hubara et al. for constructing a QNN. Namely, during forward propagation at training time, we store the real valued weights and quantize the weights and inputs to the function before performing the matrix multiplication between the two.

3.3 Quantizing the Attention Unit

Vaswani et al. model attention as queries over a set of key-value pairs. Within their original implementation they define the scaled dot product attention function as

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

yielding an architecture that looks like:

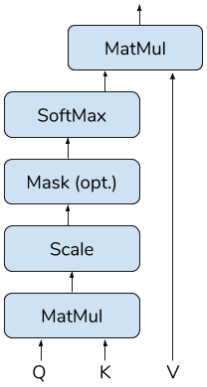

Figure 1: The scaled-dot product unit

We define a "Quantum Attention Unit" using the following observations. The scale by $\sqrt{d_k}$ is put there simply so the softmax does not overflow and thus have difficulties differentiating between values that are large in magnitude, hence we may replace it with another constant, say $2^{\text{AP2}(\sqrt{d_k})}$ where AP2(x) is defined as $\lfloor \log_2(x) \rfloor$ and can be computed by taking the most significant bit of x. This scaling factor then simply becomes a right bitshift on the values within Q. Since the input to the unit Q, K are binarized we take advantage of Hubara's gemmlowp technique (Generalized Matrix Multiplication with Low Precision) implementation that stores the result of the quantized multiplication in an accumulator of twice the bitwidth in order to avoid overflows, then clipping the product within the appropriate range to cast back into the appropriate range.

3.4 Implementation Details

For our Transformer implementation, we used the one written for the Harvard NLP Annotated Transformer document, modified as appropriate to make it compatible with the latest version of PyTorch. [Ashish Vaswani, 2017] While some inspiration was drawn from the various quantized networks posted by Hubara and Courbariaux, ultimately the implementations of any quantized functionalities (for the feed-forward and the attention unit) have been written by the author.

4 Experiments

For our experiments we worked on the IWSLT 2015 Vietnamese-English translation task. In particular, the dataset consists of the titles and subtitles of various TED and TEDX talks. There are 130,820 training examples, where each example is a sentence from one such speech. The test set we evaluated over was the 2013 testing suite, comprised of 1267 test examples.

To benchmark on this task we used BLEU score as a metric for the quality of the translations. To evaluate the BLEU scores of our models we used the sacreBLEU tool. [Post, 2018]

Reference	Fixed Width (8 bit)	Linear Quant (8 bit)
20.6	10.3	2.4

Table 1: BLEU Scores for the various models

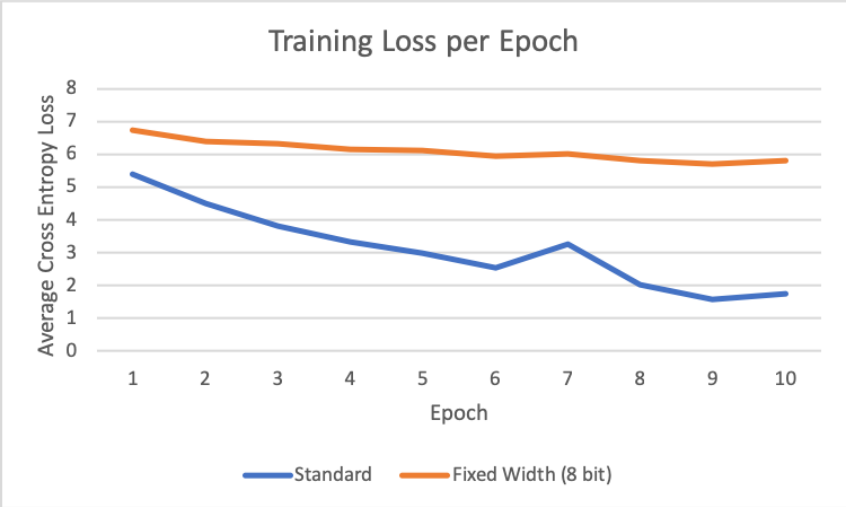

Figure 2: The training loss curve per epoch for the standard Transformer and fixed width implementations.

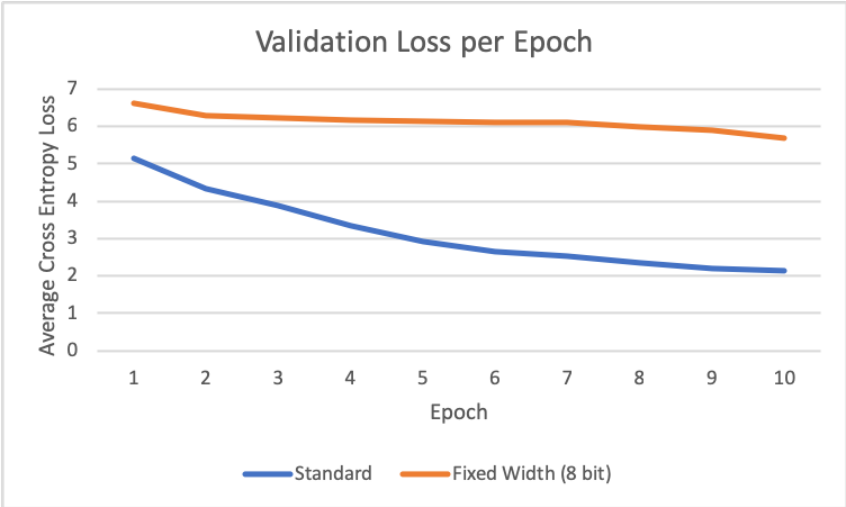

Figure 3: The validation loss curve per epoch for the standard Transformer and fixed width implementations.

References

Niki Parmar Jakob Uskoreit-Llion Jones Aidan N. Gomez Lukaz Kaiser Illia Polosukhin Ashish Vaswani, Noam Shazeer. Attention is all you need. *NIPS*, 2017.

Yoshua Bengio, Nicholas Léonard, and Aaron C. Courville. Estimating or propagating gradients through stochastic neurons for conditional computation. *CoRR*, abs/1308.3432, 2013. URL `http://arxiv.org/abs/1308.3432`.

Matthieu Courbariaux and Yoshua Bengio. Binarynet: Training deep neural networks with weights and activations constrained to +1 or -1. *CoRR*, abs/1602.02830, 2016. URL `http://arxiv.org/abs/1602.02830`.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: pre-training of deep bidirectional transformers for language understanding. *CoRR*, abs/1810.04805, 2018. URL `http://arxiv.org/abs/1810.04805`.

Andrew G. Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *CoRR*, abs/1704.04861, 2017. URL http://arxiv.org/abs/1704.04861.

Forrest N. Iandola, Matthew W. Moskewicz, Khalid Ashraf, Song Han, William J. Dally, and Kurt Keutzer. Squeezenet: Alexnet-level accuracy with 50x fewer parameters and <1mb model size. *CoRR*, abs/1602.07360, 2016. URL http://arxiv.org/abs/1602.07360.

Daniel Soudry Ran El-Yaniv Yoshua Bengio Itay Hubara, Matthieu Courbariaux. Quantized neural networks: Training neural networks with low precision weights and activations. *JMLR*, 2018.

Maximilian Lam. Word2bits - quantized word vectors. *CoRR*, abs/1803.05651, 2018. URL http://arxiv.org/abs/1803.05651.

Matt Post. A call for clarity in reporting BLEU scores. In *Proceedings of the Third Conference on Machine Translation: Research Papers*, pages 186–191. Association for Computational Linguistics, 2018. URL http://aclweb.org/anthology/W18-6319.

Mohammad Rastegari, Vicente Ordonez, Joseph Redmon, and Ali Farhadi. Xnor-net: Imagenet classification using binary convolutional neural networks. *CoRR*, abs/1603.05279, 2016. URL http://arxiv.org/abs/1603.05279.