Efficiency through Auto-Sizing: Notre Dame NLP's Submission to the WNGT 2019 Efficiency Task

Kenton Murray **Brian DuSell** **David Chiang**
Department of Computer Science and Engineering
University of Notre Dame
{kmurray4,bdusell1,dchiang}@nd.edu

Abstract

This paper describes the Notre Dame Natural Language Processing Group's (NDNLP) submission to the WNGT 2019 shared task (Hayashi et al., 2019). We investigated the impact of auto-sizing (Murray and Chiang, 2015; Murray et al., 2019) to the Transformer network (Vaswani et al., 2017) with the goal of substantially reducing the number of parameters in the model. Our method was able to eliminate more than 25% of the model's parameters while suffering a decrease of only 1.1 BLEU.

1 Introduction

The Transformer network (Vaswani et al., 2017) is a neural sequence-to-sequence model that has achieved state-of-the-art results in machine translation. However, Transformer models tend to be very large, typically consisting of hundreds of millions of parameters. As the number of parameters directly corresponds to secondary storage requirements and memory consumption during inference, using Transformer networks may be prohibitively expensive in scenarios with constrained resources. For the 2019 Workshop on Neural Generation of Text (WNGT) Efficiency shared task (Hayashi et al., 2019), the Notre Dame Natural Language Processing (NDNLP) group looked at a method of inducing sparsity in parameters called auto-sizing in order to reduce the number of parameters in the Transformer at the cost of a relatively minimal drop in performance.

Auto-sizing, first introduced by Murray and Chiang (2015), uses group regularizers to encourage parameter sparsity. When applied over neurons, it can delete neurons in a network and shrink the total number of parameters. A nice advantage of auto-sizing is that it is independent of model architecture; although we apply it to the Transformer network in this task, it can easily be applied to any other neural architecture.

NDNLP's submission to the 2019 WNGT Efficiency shared task uses a standard, recommended baseline Transformer network. Following Murray et al. (2019), we investigate the application of auto-sizing to various portions of the network. Differing from their work, the shared task used a significantly larger training dataset from WMT 2014 (Bojar et al., 2014), as well as the goal of reducing model size even if it impacted translation performance. Our best system was able to prune over 25% of the parameters, yet had a BLEU drop of only 1.1 points. This translates to over 25 million parameters pruned and saves almost 100 megabytes of disk space to store the model.

2 Auto-sizing

Auto-sizing is a method that encourages sparsity through use of a group regularizer. Whereas the most common applications of regularization will act over parameters individually, a group regularizer works over groupings of parameters. For instance, applying a sparsity inducing regularizer to a two-dimensional parameter tensor will encourage individual values to be driven to 0.0. A sparsity-inducing group regularizer will act over defined sub-structures, such as entire rows or columns, driving the entire groups to zero. Depending on model specifications, one row or column of a tensor in a neural network can correspond to one neuron in the model.

Following the discussion of Murray and Chiang (2015) and Murray et al. (2019), auto-sizing works by training a neural network while using a regularizer to prune units from the network, minimizing:

$$\mathcal{L} = -\sum_{f,e \text{ in data}} \log P(e \mid f; W) + \lambda R(\|W\|).$$

W are the parameters of the model and R is a reg-

Proceedings of the 3rd Workshop on Neural Generation and Translation (WNGT 2019), pages 297–301
Hong Kong, China, November 4, 2019.
www.aclweb.org/anthology/D19-56%2d

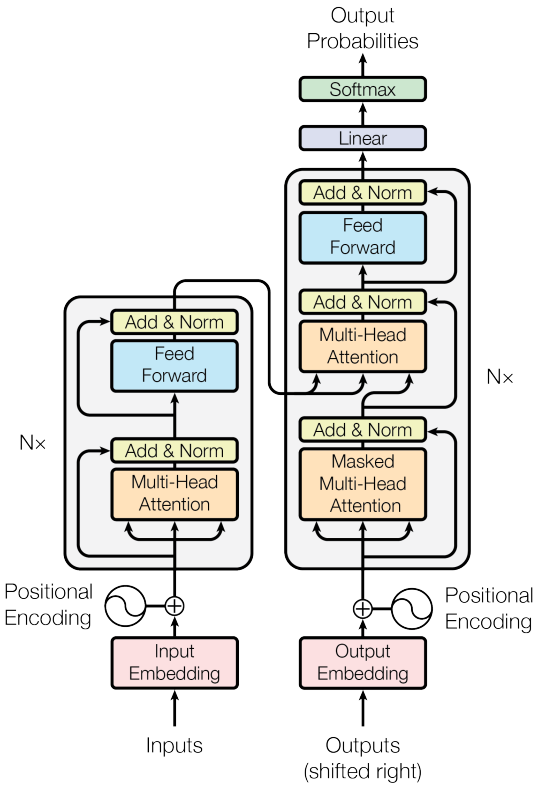

Figure 1: Architecture of the Transformer (Vaswani et al., 2017). We apply the auto-sizing method to the feed-forward (blue rectangles) and multi-head attention (orange rectangles) in all N layers of the encoder and decoder. Note that there are residual connections that can allow information and gradients to bypass any layer we are auto-sizing. Following the robustness recommendations, we instead layer norm before.

ularizer. Here, as with the previous work, we experiment with two regularizers:

$$R(W) = \sum_i \left(\sum_j W_{ij}^2 \right)^{\frac{1}{2}} \qquad (\ell_{2,1})$$

$$R(W) = \sum_i \max_j |W_{ij}| \qquad (\ell_{\infty,1})$$

The optimization is done using proximal gradient descent (Parikh and Boyd, 2014), which alternates between stochastic gradient descent steps and proximal steps:

$$W \leftarrow W - \eta \nabla \log P(e \mid f; w)$$

$$W \leftarrow \arg\min_{W'} \left(\frac{1}{2\eta} \|W - W'\|^2 + R(W') \right)$$

3 Auto-sizing the Transformer

The Transformer network (Vaswani et al., 2017) is a sequence-to-sequence model in which both the encoder and the decoder consist of stacked self-attention layers. The multi-head attention uses two affine transformations, followed by a softmax layer. Each layer has a position-wise feed-forward neural network (FFN) with a hidden layer of rectified linear units. Both the multi-head attention and the feed-forward neural network have residual connections that allow information to bypass those layers. In addition, there are also word and position embeddings. Figure 1, taken from the original paper, shows the architecture. NDNLP's submission focuses on the N stacked encoder and decoder layers.

The Transformer has demonstrated remarkable success on a variety of datasets, but it is highly over-parameterized. For example, the baseline Transformer model has more than 98 million parameters, but the English portion of the training data in this shared task has only 116 million tokens and 816 thousand types. Early NMT models such as Sutskever et al. (2014) have most of their parameters in the embedding layers, but the transformer has a larger percentage of the model in the actual encoder and decoder layers. Though the group regularizers of auto-sizing can be applied to any parameter matrix, here we focus on the parameter matrices within the encoder and decoder layers.

We note that there has been some work recently on shrinking networks through pruning. However, these differ from auto-sizing as they frequently require an arbitrary threshold and are not included during the training process. For instance, See et al. (2016) prunes networks based off a variety of thresholds and then retrains a model. Voita et al. (2019) also look at pruning, but of attention heads specifically. They do this through a relaxation of an ℓ_0 regularizer in order to make it differentiable. This allows them to not need to use a proximal step. This method too starts with pre-trained model and then continues training. Michel et al. (2019) also look at pruning attention heads in the transformer. However, they too use thresholding, but only apply it at test time. Auto-sizing does not require a thresholding value, nor does it require a pre-trained model.

Of particular interest are the large, position-wise feed-forward networks in each encoder and decoder layer:

$$\mathrm{FFN}(x) = W_2(\max(0, W_1 x + b_1)) + b_2.$$

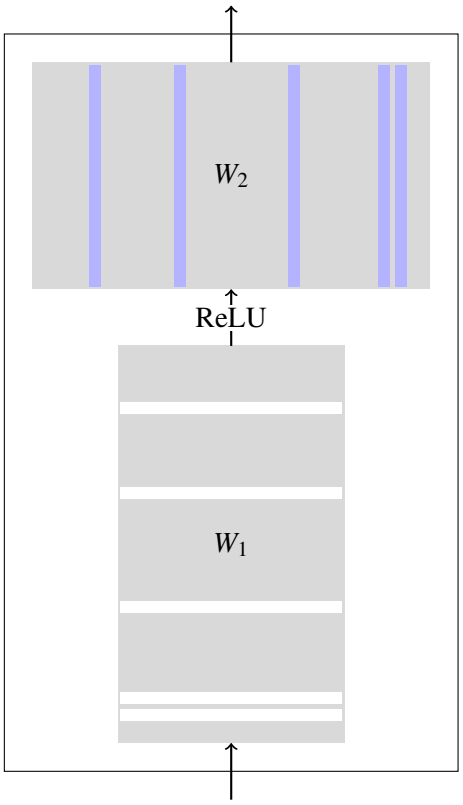

Figure 2: Auto-sizing FFN network. For a row in the parameter matrix W_1 that has been driven completely to 0.0 (shown in white), the corresponding column in W_2 (shown in blue) no longer has any impact on the model. Both the column and the row can be deleted, thereby shrinking the model.

W_1 and W_2 are two large affine transformations that take inputs from D dimensions to $4D$, then project them back to D again. These layers make use of rectified linear unit activations, which were the focus of auto-sizing in the work of Murray and Chiang (2015). No theory or intuition is given as to why this value of $4D$ should be used.

Following (Murray et al., 2019), we apply the auto-sizing method to the Transformer network, focusing on the two largest components, the feed-forward layers and the multi-head attentions (blue and orange rectangles in Figure 1). Remember that since there are residual connections allowing information to bypass the layers we are auto-sizing, information can still flow through the network even if the regularizer drives all the neurons in a layer to zero – effectively pruning out an entire layer.

4 Experiments

All of our models are trained using the fairseq implementation of the Transformer (Gehring et al., 2017).[1] For the regularizers used in auto-sizing, we make use of an open-source, proximal gradient toolkit implemented in PyTorch[2] (Murray et al., 2019). For each mini-batch update, the stochastic gradient descent step is handled with a standard PyTorch forward-backward call. Then the proximal step is applied to parameter matrices.

4.1 Settings

We used the originally proposed transformer architecture – with six encoder and six decoder layers. Our model dimension was 512 and we used 8 attention heads. The feed-forward network sub-components were of size 2048. All of our systems were run using subword units (BPE) with 32,000 merge operations on concatenated source and target training data (Sennrich and Haddow, 2016). We clip norms at 0.1, use label smoothed cross-entropy with value 0.1, and an early stopping criterion when the learning rate is smaller than 10^{-5}. We used the Adam optimizer (Kingma and Ba, 2015), a learning rate of 10^{-4}, and dropout of 0.1. Following recommendations in the fairseq and tensor2tensor (Vaswani et al., 2018) code bases, we apply layer normalization before a sub-component as opposed to after. At test time, we decoded using a beam of 5 with length normalization (Boulanger-Lewandowski et al., 2013) and evaluate using case-sensitive, tokenized BLEU (Papineni et al., 2002).

For the auto-sizing experiments, we looked at both $\ell_{2,1}$ and $\ell_{\infty,1}$ regularizers. We experimented over a range of regularizer coefficient strengths, λ, that control how large the proximal gradient step will be. Similar to Murray and Chiang (2015), but differing from Alvarez and Salzmann (2016), we use one value of λ for all parameter matrices in the network. We note that different regularization coefficient values are suited for different types or regularizers. Additionally, all of our experiments use the same batch size, which is also related to λ.

4.2 Auto-sizing sub-components

We applied auto-sizing to the sub-components of the encoder and decoder layers, without touching the word or positional embeddings. Recall from

[1] https://github.com/pytorch/fairseq
[2] https://github.com/KentonMurray/ProxGradPytorch

System	Disk Size	Number of Parameters	newstest2014	newstest2015
Baseline	375M	98.2M	25.3	27.9
All $\ell_{2,1} = 0.1$	345M	90.2M	21.6	24.1
Encoder $\ell_{2,1} = 0.1$	341M	89.4M	23.2	25.5
Encoder $\ell_{2,1} = 1.0$	327M	85.7M	22.1	24.5
FFN $\ell_{2,1} = 0.1$	326M	85.2M	24.1	26.4
FFN $\ell_{2,1} = 1.0$	279M	73.1M	24.0	26.8
FFN $\ell_{2,1} = 10.0$	279M	73.1M	23.9	26.5
FFN $\ell_{\infty,1} = 100.0$	327M	73.1M	23.8	26.0

Table 1: Comparison of BLEU scores and model sizes on newstest2014 and newstest2015. Applying auto-sizing to the feed-forward neural network sub-components of the transformer resulted in the most amount of pruning while still maintaining good BLEU scores.

Figure 1, that each layer has multi-head attention and feed-forward network sub-components. In turn, each multi-head attention sub-component is comprised of two parameter matrices. Similarly, each feed-forward network has two parameter matrices, W_1 and W_2. We looked at three main experimental configurations:

- All: Auto-sizing is applied to every multi-head attention and feed-forward network sub-component in every layer of the encoder and decoder.
- Encoder: As with All, auto-sizing is applied to both multi-head attention and feed-forward network sub-components, but only in the encoder layers. The decoder remains the same.
- FFN: Auto-sizing applied only to the feed-forward network sub-components W_1 and W_2, but not to the multi-head portions. This too is applied to both the encoder and decoder.

4.3 Results

Our results are presented in Table 1. The baseline system has 98.2 million parameters and a BLEU score of 29.7. It takes up 375 megabytes on disk. Our systems that applied auto-sizing only to the feed-forward network sub-components of the transformer network maintained the best BLEU scores while also pruning out the most parameters of the model. Overall, our best system used $\ell_{2,1} = 1.0$ regularization for auto-sizing and left 73.1 million parameters remaining. On disk, the model takes 279 megabytes to store – roughly 100 megabytes less than the baseline. The performance drop compared to the baseline is 1.1 BLEU points, but the model is over 25% smaller.

Applying auto-sizing to the multi-head attention and feed-forward network sub-components of *only* the encoder also pruned a substantial amount of parameters. Though this too resulted in a smaller model on disk, the BLEU scores were worse than auto-sizing just the feed-forward sub-components. Auto-sizing the multi-head attention and feed-forward network sub-components of both the encoder *and* decoder actually resulted in a larger model than the encoder only, but with a lower BLEU score. Overall, our results suggest that the attention portion of the transformer network is more important for model performance than the feed-forward networks in each layer.

5 Conclusion

In this paper, we have investigated the impact of using auto-sizing on the transformer network of the 2019 WNGT efficiency task. We were able to delete more than 25% of the parameters in the model while only suffering a modest BLEU drop. In particular, focusing on the parameter matrices of the feed-forward networks in every layer of the encoder and decoder yielded the smallest models that still performed well.

A nice aspect of our proposed method is that the proximal gradient step of auto-sizing can be applied to a wide variety of parameter matrices. Whereas for the transformer, the largest impact was on feed-forward networks within a layer, should a new architecture emerge in the future, auto-sizing can be easily adapted to the trainable parameters.

Overall, NDNLP's submission has shown that auto-sizing is a flexible framework for pruning parameters in a large NMT system. With an aggressive regularization scheme, large portions of

the model can be deleted with only a modest impact on BLEU scores. This in turn yields a much smaller model on disk and at run-time.

Acknowledgements

This research was supported in part by University of Southern California, subcontract 67108176 under DARPA contract HR0011-15-C-0115.

References

Jose M Alvarez and Mathieu Salzmann. 2016. Learning the number of neurons in deep networks. In *Advances in Neural Information Processing Systems*, pages 2270–2278.

Ondrej Bojar, Christian Buck, Christian Federmann, Barry Haddow, Philipp Koehn, Johannes Leveling, Christof Monz, Pavel Pecina, Matt Post, Herve Saint-Amand, et al. 2014. Findings of the 2014 workshop on statistical machine translation. In *Proceedings of the ninth workshop on statistical machine translation*, pages 12–58.

Nicolas Boulanger-Lewandowski, Yoshua Bengio, and Pascal Vincent. 2013. Audio chord recognition with recurrent neural networks. In *Proc. International Society for Music Information Retrieval*, pages 335–340.

Jonas Gehring, Michael Auli, David Grangier, Denis Yarats, and Yann N. Dauphin. 2017. Convolutional Sequence to Sequence Learning. In *Proc. ICML*.

Hiroaki Hayashi, Yusuke Oda, Alexandra Birch, Ioannis Constas, Andrew Finch, Minh-Thang Luong, Graham Neubig, and Katsuhito Sudoh. 2019. Findings of the third workshop on neural generation and translation. In *Proceedings of the Third Workshop on Neural Generation and Translation*.

Diederik P. Kingma and Jimmy Lei Ba. 2015. Adam: A method for stochastic optimization. In *Proc. ICLR*.

Paul Michel, Omer Levy, and Graham Neubig. 2019. Are sixteen heads really better than one? *Advances in Neural Information Processing Systems*.

Kenton Murray and David Chiang. 2015. Auto-sizing neural networks: With applications to *n*-gram language models. In *Proc. EMNLP*.

Kenton Murray, Jeffery Kinnison, Toan Q. Nguyen, Walter Scheirer, and David Chiang. 2019. Auto-sizing the transformer network: Improving speed, efficiency, and performance for low-resource machine translation. In *Proceedings of the Third Workshop on Neural Generation and Translation*.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. BLEU: a method for automatic evaluation of machine translation. In *Proc. ACL*, pages 311–318.

Neal Parikh and Stephen Boyd. 2014. Proximal algorithms. *Foundations and Trends in Optimization*, 1(3):123–231.

Abigail See, Minh-Thang Luong, and Christopher D Manning. 2016. Compression of neural machine translation models via pruning. In *Proceedings of The 20th SIGNLL Conference on Computational Natural Language Learning*, pages 291–301.

Rico Sennrich and Barry Haddow. 2016. Linguistic input features improve neural machine translation. In *Proc. First Conference on Machine Translation: Volume 1, Research Papers*, volume 1, pages 83–91.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *Advances in Neural Information Processing Systems*, pages 3104–3112.

Ashish Vaswani, Samy Bengio, Eugene Brevdo, Francois Chollet, Aidan N. Gomez, Stephan Gouws, Llion Jones, Łukasz Kaiser, Nal Kalchbrenner, Niki Parmar, Ryan Sepassi, Noam Shazeer, and Jakob Uszkoreit. 2018. Tensor2tensor for neural machine translation. *CoRR*, abs/1803.07416.

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

Elena Voita, David Talbot, Fedor Moiseev, Rico Sennrich, and Ivan Titov. 2019. Analyzing multi-head self-attention: Specialized heads do the heavy lifting, the rest can be pruned. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5797–5808, Florence, Italy. Association for Computational Linguistics.