Scalable Language Modeling: WikiText-103 on a Single GPU in 12 hours

Extended Abstract

Stephen Merity*
Salesforce Research
Palo Alto, CA
smerity@salesforce.com

Nitish Shirish Keskar*
Salesforce Research
Palo Alto, CA
nkeskar@salesforce.com

James Bradbury
Salesforce Research
Palo Alto, CA
james.bradbury@salesforce.com

Richard Socher
Salesforce Research
Palo Alto, CA
rsocher@salesforce.com

ABSTRACT

Word-level language modeling (WLM) is one the foundational tasks of unsupervised natural language processing. Most modern architectures for WLM use several LSTM layers, followed by a softmax layer. Even with larger batch sizes and a multi-GPU setup, training of these networks on large-vocabulary corpora is slow due to increased computation involving the softmax and the high cost of recurrence computation. We propose a model architecture and training strategy that enables us to achieve state-of-the-art performance on the WikiText-103 data set using a single GPU while being substantially faster than an NVIDIA cuDNN LSTM-based model by utilizing the Quasi-Recurrent Neural Network (QRNN), an adaptive softmax with weight tying, and longer sequences within batches.

ACM Reference Format:
Stephen Merity, Nitish Shirish Keskar*, James Bradbury, and Richard Socher. 2018. Scalable Language Modeling: WikiText-103 on a Single GPU in 12 hours: Extended Abstract. In *Proceedings of SysML Conference (SYSML'18).* ACM, New York, NY, USA, Article 4, 2 pages. https://doi.org/10.475/123_4

1 INTRODUCTION

Language modeling (LM) is one of the foundational tasks of natural language processing. The task involves predicting the $(n+1)^{th}$ token in a sequence given the n preceding tokens, these tokens may be words, characters or sub-words. Trained LMs are useful in many models including speech recognition [17], machine translation [10], natural language generation [12], and generating token embeddings. In this paper, we focus on the specific task of word-level language modeling (WLM) where the sequence is composed of tokens in the form of words. Currently, most state-of-the-art models for WLM use an embedding layer to map the discrete tokens into a continuous vector space, followed by recurrent or convolutional layers, and a final softmax layer which maps the latent representation to the vocabulary space. For large corpora, such as the WikiText-103 [13] or the One Billion Word Benchmark [2], these networks are slow to train due to the number of tokens, the size of the vocabulary, the speed of recurrent neural models, and the prohibitive costs of a softmax over hundreds of thousands words.

To enable faster training of large-scale corpora, many approaches have been explored. A popular approach, especially in the computer vision field, is data parallelism through increased batch sizes [4, 16]. In this paper, we take an orthogonal approach and reduce training time through architecture design.

2 MODEL ARCHITECTURE

Our model is based on the AWD-QRNN proposed in [11]. By using the Quasi-Recurrent Neural Network (QRNN) [1] instead of the LSTM, we avoid costly recurrence computation.

The key to the QRNN architecture is that recurrence within the RNN does not involve a matrix multiplication and is instead performed as an elementwise operation. This avoids the sequential bottleneck that traditionally accompanies RNN architecture, with the QRNN's recurrence operation taking less time than the application of dropout to the QRNN's input. This greatly decreases training time, especially when larger sequence lengths are chosen. In our model, we use 4 QRNN layers with 2500 nodes in each layer. As in [11], we use a 400-dimensional embedding layer with tied softmax weights.

To address large vocabulary sizes, which can result in impractically slow models, we use adaptive softmax [5], which uses a hierarchy determined by word frequency to reduce computation time, modified to allow weight tying [8, 15]. Similar strategies for improved softmax training speed have been proposed earlier [7, 14].

To our knowledge, this is the first work to use weight tying in conjunction with the adaptive softmax. Weight tying has been shown to substantially improve the prediction of rare words. Given the adaptive softmax is specifically targeted toward large vocabularies, with most of the words being infrequent due to Zipf's Law, we suggest this should be a standard tactic.

*Equal Contribution.

SYSML'18, February, 2018, Palo Alto, CA USA

ACM ISBN 123-4567-24-567/08/06…$15.00
https://doi.org/10.475/123_4

Model	Val	Test
Grave et al. [6]	–	48.7
Dauphin et al. [3], 1 GPU	–	44.9
Dauphin et al. [3], 4 GPUs	–	37.2
Proposed, 1 GPU	32.0	33.0

Table 1: Perplexity on WikiText-103.

Model	Time per batch
LSTM	726ms
QRNN	233ms

Table 2: Mini-batch timings during training.

3 EXPERIMENTAL RESULTS

While the speed gains should be applicable to many large corpora, we validate our model against WikiText-103. The WikiText-103 dataset contains 103 million tokens and a vocabulary of 267,735. As opposed to the One Billion Word Benchmark, which uses randomly shuffled sentences, the WikiText-103 dataset uses whole articles from Wikipedia, allowing for long term dependencies.

In order to increase concurrency in training, we propose not only increasing batch sizes but also sequence lengths. Unlike tasks like neural machine translation, wherein the sequence length is determined by the size of the sentence, WLM allows for training of sequences with arbitrary length. Traditionally increasing sequence lengths has been known to make training both slower and more difficult. In our experiments with sequence lengths of up to 200, we found no tangible difficulties for backpropagation and experience a minimal loss in speed due to the QRNN. We use Adam [9] with a learning rate of 1×10^{-3} as the optimizer and train for 14 epochs. We reduce the learning rate by 10 on epoch 12. Our model consists of an embedding layer of size 400, 4 layers of 2500 nodes and an adaptive softmax layer. We train with a batch size of 60 and a sequence length of 140. To avoid over-fitting, we employ the regularization strategies proposed in [11] including variational dropout, random sequence lengths, and L_2-norm decay. The values for the model hyperparameters were tuned only coarsely.

The model trains at 2980 seconds per epoch on the NVIDIA V100 and 5460 seconds per epoch on the NVIDIA P100. This is a 3.1 times speed-up over an NVIDIA cuDNN LSTM baseline which uses the same model hyperparameters. We report single model perplexity and per batch timing in Table 1.

4 DISCUSSION

Scaling is traditionally achieved by increasing the number of examples per batch, seen frequently in the computer vision community. Our results above show that using the QRNN and increasing the sequence length allows for increased concurrency even with small batch sizes. With such a high utilization on a single GPU, we believe this is a promising strategy for achieving strong speed-ups in language modeling within a multi-GPU setting. We also expect further reductions in training time by using reduced precision computation (FP16) on the NVIDIA Pascal and Volta GPU architectures.

While working at the word level has many advantages, the softmax overhead could also be reduced by splitting the existing word level vocabulary to a subword representation. This would substantially reduce the size of the vocabulary, improving both the number of parameters used by the model's word embeddings as well as decreasing the computation time for each softmax call during training. The impact that such a tactic would have upon weight tying is unknown however as weight tying has traditionally only been performed upon a word level vocabulary.

5 CONCLUSION

In this work, we describe our tactics for reducing trainings times and improving perplexity for large scale language modeling corpora. Using the AWD-QRNN as a foundation, we discuss our architectural choices for using the QRNN instead of the LSTM and adaptive softmax with weight tying. We also discuss increasing concurrency in not just the size of the batches but increasing We also discuss increasing concurrency in each batch by increasing the sequence length of the batches themselves. Using this approach we reduce our per-epoch time substantially and achieve a new state-of-the-art on WikiText-103 despite training for 14 epochs, a total time of only 12 hours.

REFERENCES

[1] J. Bradbury, S. Merity, C. Xiong, and R. Socher. 2016. Quasi-Recurrent Neural Networks. *arXiv preprint arXiv:1611.01576* (2016).

[2] C. Chelba, T. Mikolov, M. Schuster, Q. Ge, T. Brants, P. Koehn, and T. Robinson. 2013. *One Billion Word Benchmark for Measuring Progress in Statistical Language Modeling.* Technical Report. Google. http://arxiv.org/abs/1312.3005

[3] Y. Dauphin, A. Fan, M. Auli, and D. Grangier. 2016. Language modeling with Gated Convolutional Networks. *arXiv preprint arXiv:1612.08083* (2016).

[4] P. Goyal, P. Dollár, R. Girshick, P. Noordhuis, L. Wesolowski, A. Kyrola, A. Tulloch, Y. Jia, and K. He. 2017. Accurate, Large Minibatch SGD: Training ImageNet in 1 Hour. *arXiv preprint arXiv:1706.02677* (2017).

[5] E. Grave, A. Joulin, M. Cissé, D. Grangier, and H. Jégou. 2016. Efficient softmax approximation for GPUs. *arXiv preprint arXiv:1609.04309* (2016).

[6] E. Grave, A. Joulin, and N. Usunier. 2016. Improving neural language models with a continuous cache. *arXiv preprint arXiv:1612.04426* (2016).

[7] M. Gutmann and A. Hyvärinen. 2010. Noise-contrastive estimation: A new estimation principle for unnormalized statistical models. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*. 297–304.

[8] H. Inan, K. Khosravi, and R. Socher. 2016. Tying Word Vectors and Word Classifiers: A Loss Framework for Language Modeling. *arXiv preprint arXiv:1611.01462* (2016).

[9] D. Kingma and J. Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014).

[10] P. Koehn. 2009. *Statistical Machine Translation.* Cambridge University Press.

[11] S. Merity, N. Keskar, and R. Socher. 2017. Regularizing and Optimizing LSTM Language Models. *arXiv preprint arXiv:1708.02182* (2017).

[12] S. Merity, B. McCann, and R. Socher. 2017. Revisiting Activation Regularization for Language RNNs. *arXiv preprint arXiv:1708.01009* (2017).

[13] S. Merity, C. Xiong, J. Bradbury, and R. Socher. 2016. Pointer Sentinel Mixture Models. *arXiv preprint arXiv:1609.07843* (2016).

[14] F. Morin and Y. Bengio. 2005. Hierarchical Probabilistic Neural Network Language Model. In *Aistats*, Vol. 5. 246–252.

[15] O. Press and L. Wolf. 2016. Using the output embedding to improve language models. *arXiv preprint arXiv:1608.05859* (2016).

[16] Y. You, I. Gitman, and B. Ginsburg. 2017. Scaling SGD batch size to 32k for imagenet training. *arXiv preprint arXiv:1708.03888* (2017).

[17] D. Yu and L. Deng. 2014. *Automatic speech recognition: A deep learning approach.* Springer.