Scaling Neural Machine Translation

Myle Ott[△] **Sergey Edunov**[▽] **David Grangier**[▽] **Michael Auli**[▽]
Facebook AI Research
[△]New York, NY [▽]Menlo Park, CA

Abstract

Sequence to sequence learning models still require several days to reach state of the art performance on large benchmark datasets using a single machine. This paper shows that reduced precision and large batch training can speedup training by nearly 5x on a single 8-GPU machine with careful tuning and implementation.[1] On WMT'14 English-German translation, we match the accuracy of Vaswani et al. (2017) in under 5 hours when training on 8 GPUs and we obtain a new state of the art of 29.3 BLEU after training for 91 minutes on 128 GPUs. We further improve these results to 29.8 BLEU by training on the much larger Paracrawl dataset.

1 Introduction

Neural Machine Translation (NMT) has seen impressive progress in the recent years with the introduction of ever more efficient architectures (Bahdanau et al., 2015; Gehring et al., 2017; Vaswani et al., 2017). Similar sequence-to-sequence models are also applied to other natural language processing tasks, such as abstractive summarization (See et al., 2017; Paulus et al., 2018) and dialog (Sordoni et al., 2015; Serban et al., 2017; Dusek and Jurcícek, 2016).

Currently, training state-of-the-art models on large datasets is computationally intensive and can require several days on a machine with 8 high-end graphics processing units (GPUs). Scaling training to multiple machines enables faster experimental turn-around but also introduces new challenges: How do we maintain efficiency in a distributed setup when some batches process faster than others? How do larger batch sizes affect generalization performance? The former is specific to multi-machine training but the latter is a preview of the challenges that even users of commodity hardware are likely to face soon, assuming that hardware improves at the rapid rate it has thus far.

In this paper, we first explore approaches to improve training efficiency on a single machine. By training with reduced precision we decrease training time by 65% with no effect on accuracy. Next, we assess the effect of dramatically increasing the batch size from 25k to over 400k tokens, a necessary condition for large scale parallelization with synchronous stochastic gradient descent (SGD). We implement this on a single machine by accumulating gradients from several batches before each update. We find that by training with large batches and by increasing the learning rate we can further reduce training time by 40% on a single machine and by 90% in a distributed 16-machine setup.

Our improvements enable training a WMT'14 En-De model to the same accuracy as Vaswani et al. (2017) in just 37 minutes on 128 GPUs. When training to full convergence, we achieve a new state of the art of 29.3 BLEU in 91 minutes. These scalability efforts also enable us to train models on much larger datasets. We show that we can reach 29.8 BLEU on the same test set in less than 10 hours when trained on a combined corpus of WMT and Paracrawl data containing ∼150M sentence pairs. Similarly, on the WMT'14 En-Fr task we obtain a state of the art BLEU of 43.2 in 8.5 hours on 128 GPUs.

2 Related Work

Previous research considered efficient training and inference with reduced numerical precision for neural networks (Simard and Graf, 1993; Courbariaux et al., 2015; Sa et al., 2018). Our work relies on half-precision floating point computation and follows the guidelines of Narang et al. (2018) to adjust the scale of the loss to avoid underflow or overflow errors in gradient computations.

Distributed training of neural networks follows

[1]Our implementation is available at: https://github.com/pytorch/fairseq

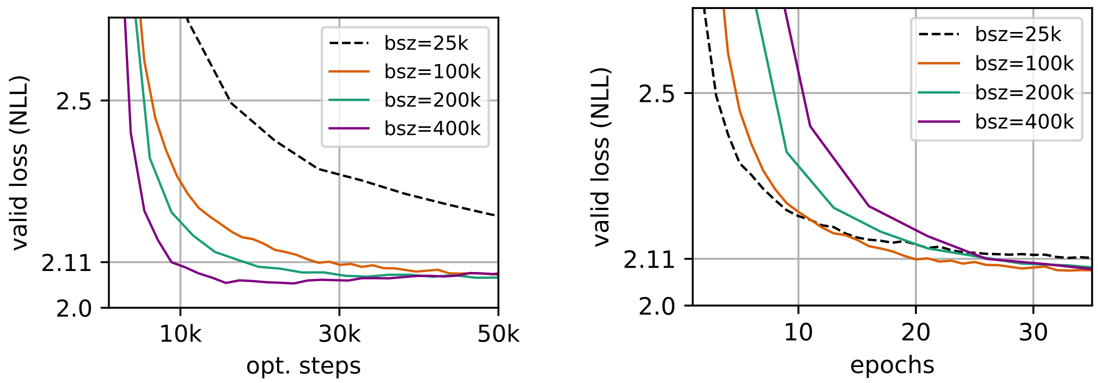

Figure 1: Validation loss for Transformer model trained with varying batch sizes (bsz) as a function of optimization steps (left) and epochs (right). Training with large batches is less data-efficient, but can be parallelized. Batch sizes given in number of target tokens excluding padding. *WMT'14 En-De, newstest13*.

two main strategies: (i) *model parallel* evaluates different model layers on different workers (Coates et al., 2013) and (ii) *data parallel* keeps a copy of the model on each worker but distributes different batches to different machines (Dean et al., 2012). We rely on the second scheme using synchronous SGD, which has recently been deemed more efficient (Chen et al., 2016). Synchronous SGD distributes the computation of gradients over multiple machines and then performs a synchronized update of the model weights. Large neural machine translation systems have been recently trained with this algorithm with success (Dean, 2017; Chen et al., 2018).

3 Experimental Setup

3.1 Datasets and Evaluation

We run experiments on two language pairs, English to German (En–De) and English to French (En–Fr). For En–De we replicate the setup of Vaswani et al. (2017) which relies on WMT'16 for training with 4.5m sentence pairs, we validate on newstest13 and test on newstest14. The 32k vocabulary is based on a joint source and target byte pair encoding (BPE; Sennrich et al. 2016). For En–Fr, we train on WMT'14 and borrow the setup of Gehring et al. (2017) with 36m training sentence pairs. We use newstest12+13 for validation and newstest14 for test. The 40k vocabulary is based on a joint source and target BPE.

We also experiment with scaling training beyond 36m sentence pairs by using data from the Paracrawl corpus (ParaCrawl, 2018). This dataset is extremely large with more than 4.5B pairs for En–De and more than 4.2B pairs for En–Fr. We rely on the BPE vocabulary built on WMT data for each pair and explore filtering this noisy dataset in Section 4.4.

We measure case-sensitive tokenized BLEU with multi-bleu (Hoang et al., 2006) and detokenized BLEU with SacreBLEU[2] (Post, 2018). All results use beam search with a beam width of 4 and length penalty of 0.6, following Vaswani et al. 2017.

3.2 Models and Hyperparameters

We reimplemented the transformer model (Vaswani et al., 2017) in PyTorch using the `fairseq-py` toolkit (Edunov et al., 2017). All experiments are based on the "big" transformer model with 6 blocks in the encoder and decoder networks (word representations of size 1024, feed-forward layers with inner dimension 4,096, 16 attention heads). Dropout is set to 0.3 for En-De and 0.1 for En-Fr. Checkpoint averaging is not used, except where specified otherwise.

Models are optimized with Adam (Kingma and Ba, 2015) using $\beta_1 = 0.9$, $\beta_2 = 0.98$, and $\epsilon = 1e{-}8$ and we use the same learning rate schedule as Vaswani et al. (2017): the learning rate increases linearly for 4,000 steps to $5e{-}4$, after which it is decayed proportionally to the inverse square root of the number of steps. We use label smoothing with 0.1 weight for the uniform prior distribution over the vocabulary (Szegedy et al.,

[2]SacreBLEU hash: `BLEU+case.mixed+lang.en-{de,fr}+numrefs.1+smooth.exp+test.wmt14/full+tok.13a+version.1.2.9`

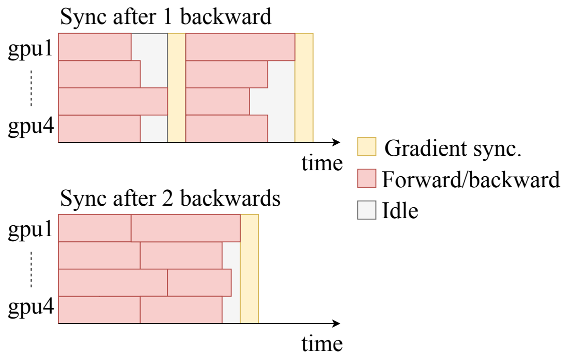

Figure 2: Accumulating gradients over multiple forward/backwards speeds up training by: (i) reducing communication, and (ii) saving idle time by reducing variance in workload between GPUs.

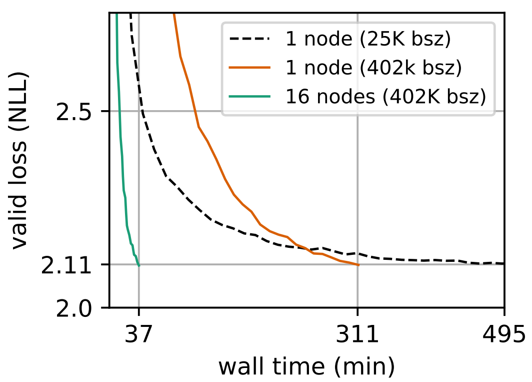

Figure 3: Validation loss (negative log likelihood on newstest13) versus training time on 1 vs 16 nodes.

2015; Pereyra et al., 2017).

All experiments run on DGX-1 nodes with 8 Nvidia V100 GPUs interconnected by Infiniband. We use the NCCL2 library and `torch.distributed` for inter-GPU communication.

4 Experiments and Results

4.1 Half-Precision Training

Recent GPUs enable efficient half precision floating point (FP) computation. This however requires scaling the loss to achieve comparable model quality (Narang et al., 2018). We first compare a baseline transformer model trained on 8 GPUs with 32-bit floating point (Our reimplementation) to the same model trained with 16-bit floating point (`16-bit`). Table 1 reports training speed of various setups to reach validation perplexity 4.32 and shows that `16-bit` results in a 2.9x speedup.

4.2 Training with Larger Batches

Large batches are a prerequisite for distributed synchronous SGD, since it averages the gradients over all workers and thus the effective batch size is the sum of the sizes of all batches seen by the workers.

Figure 1 shows that bigger batches result in slower initial convergence when measured in terms of epochs (i.e. passes over the training set). However, when looking at the number of weight updates (i.e. optimization steps) large batches converge faster (Hoffer et al., 2017). These results support parallelization since the number of steps define the number of synchronization points for synchronous SGD.

Training with large batches is also possible on a single machine regardless of the number of GPUs or amount of available memory; one simply iterates over multiple batches and accumulates the resulting gradients before committing a weight update. This has the added benefit of reducing communication and reducing the variance in workload between different workers (see Figure 2), leading to a 36% increase in tokens/sec (Table 1, `cumul`).

Similar to Goyal et al. (2017) and Smith et al. (2018) we find that increasing the learning rate for large batches shortens training time, even on a single node (`2x lr`). Table 1 reports our speed improvements due to reduced precision, larger batches and learning rate increase. Overall, we reduce training time from 1,429 min to 294 min to reach the same perplexity on the same hardware (8x Nvidia V100), i.e. a 4.9x speedup.

Memory Efficiency Reduced precision also decreases memory consumption, allowing for larger batches per GPU. We switch from a maximum of 3.5k tokens per GPU to a maximum of 5k tokens per GPU and obtain an additional 5% speedup (cf. Table 1; `2x lr` vs. `5k tkn/gpu`).

Parallel Training Large batch training can be easily parallelized. We run our previous 1-node experiment over 16 nodes of 8 GPUs each (Nvidia V100). Table 1 shows that with parallel training over 16 nodes we can further reduce training time from 311 minutes to just 37 minutes (cf. Table 1; `2x lr` vs. `16 nodes`). We illustrate this comparison in Figure 3.

model	# gpu	bsz	cumul	BLEU	updates	tkn/sec	time	speedup
Vaswani et al. (2017)	8×P100	25k	1	26.4	300k	~25k	~5,000	–
Our reimplementation	8×V100	25k	1	26.4	192k	54k	1,429	reference
+ `16-bit`	8	25k	1	26.7	193k	143k	495	2.9x
+ `cumul`	8	402k	16	26.7	13.7k	195k	447	3.2x
+ `2x lr`	8	402k	16	26.5	9.6k	196k	311	4.6x
+ `5k tkn/gpu`	8	365k	10	26.5	10.3k	202k	294	4.9x
16 nodes (from +`2x lr`)	128	402k	1	26.5	9.5k	1.53M	37	38.6x

Table 1: Training time (min) for reduced precision (`16-bit`), cumulating gradients over multiple backwards (`cumul`), increasing learning rate (`2x lr`) and computing each forward/backward with more data due to memory savings (`5k tkn/gpu`). Average time (excl. validation and saving models) over 3 random seeds to reach validation perplexity of 4.32 (2.11 NLL). Cumul=16 means a weight update after accumulating gradients for 16 backward computations, simulating training on 16 nodes. *WMT'14 En-De, newstest13*.

	En–De	En–Fr
Gehring et al. (2017)	25.2	40.5
Vaswani et al. (2017)	28.4	41.0
Ahmed et al. (2017)	28.9	41.4
Shaw et al. (2018)	29.2	41.5
Our result	**29.3**	**43.2**
16-node training time	*91 min*	*515 min*

Table 2: Test BLEU (*newstest14*). En–De trained on WMT-16 (Vaswani et al., 2017). En–Fr trained on WMT-14 (Gehring et al., 2017).

Train set	En–De	En–Fr
WMT only	29.3	**43.2**
detok. SacreBLEU	*28.6*	*41.4*
16-node training time	*91 min*	*512 min*
WMT + Paracrawl	**29.8**	42.1
detok. SacreBLEU	*29.3*	*40.9*
16-node training time	*539 min*	*794 min*

Table 3: Test BLEU (*newstest14*) when training with WMT+Paracrawl data.

4.3 Results with WMT Training Data

We report results on newstest14 for English-to-German (En-De) and English-to-French (En-Fr). For En-De, we train on the filtered version of WMT-16 from Vaswani et al. (2017). For En-Fr, we follow the setup of Gehring et al. (2017). In both cases, we train a "big" transformer on 16 nodes and average model parameters from the last 10 checkpoints (Vaswani et al., 2017). Table 2 reports 29.3 BLEU for En-De in 1h31min and 43.2 BLEU for En-Fr in 8h32min. We therefore establish a new state-of-the-art for both datasets, excluding settings with additional training data (Kutylowski, 2018). In contrast to Table 1, Table 2 reports times to convergence, not times to a specific validation likelihood.

4.4 Results with WMT & Paracrawl Training

Fast parallel training lets us explore training over larger datasets. We consider Paracrawl (ParaCrawl, 2018), a recent dataset of more than 4B parallel sentences for each language pair. Previous work on Paracrawl considered training only on filtered subsets of less than 30M pairs (Xu and Koehn, 2017). We also filter Paracrawl, in particular, we remove sentence-pairs with a source/target length ratio exceeding 1.5 and with sentences of more than 250 words. We also remove pairs for which the source and target are copies (Ott et al., 2018). On En–De, this brings the set from 4.6B to 700m. We then train a En–De model on a clean dataset (WMT'14 news commentary) to score the remaining 700m sentence pairs. We retain the 140m pairs with best average token log-likelihood. Figure 4 shows models for which we sampled WMT and filtered Paracrawl at different ratios during training. The model with 1:1 ratio performs best on valid. To train an En–Fr model, we filter the data to 129m pairs using the same procedure and use a sampling ratio of 3:1.

Overall, Paracrawl improves BLEU on En–De to 29.8 (Table 3) but it is not beneficial for En–Fr, achieving just 42.1 vs. 43.2 BLEU for our baseline.

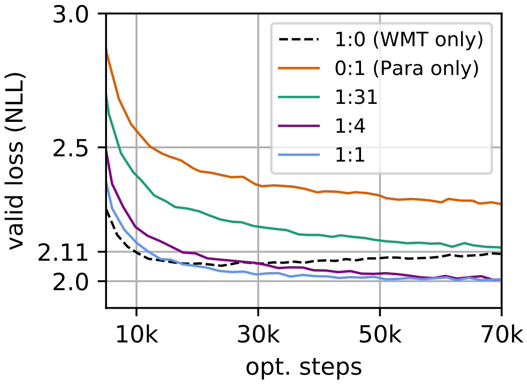

Figure 4: Validation loss when training on Paracrawl+WMT with varying sampling ratios. 1:4 means sampling 4 Paracrawl sentences for every WMT sentence. *WMT'14 En-De, newstest13*.

5 Conclusions

We explored how to train state-of-the-art NMT models on large scale parallel hardware. We investigated lower precision computation, very large batch sizes (up to 400k tokens), and larger learning rates. Our careful implementation speeds up the training of a big transformer model (Vaswani et al., 2017) by nearly 5x on one machine with 8 GPUs.

We improve the state-of-the-art for WMT'14 En-Fr to 43.2 vs. 41.5 for Shaw et al. (2018), training in less than 9 hours on 128 GPUs. On WMT'14 En-De, we report 29.3 BLEU vs 29.2 for Shaw et al. (2018) on the same setup, training our model in 91 minutes on 128 GPUs. BLEU is further improved to 29.8 by scaling the training set with Paracrawl data.

Overall, our work shows that future hardware will enable training times for large NMT systems that are comparable to phrase-based systems (Koehn et al., 2007). We note that multi-node parallelization still incurs a significant overhead: 16-node training is only ~8x faster than 1-node training. Future work may consider better batching and communication strategies.

References

Karim Ahmed, Nitish Shirish Keskar, and Richard Socher. 2017. Weighted transformer network for machine translation. *arxiv*, 1711.02132.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *Proc. of ICLR*.

Jianmin Chen, Rajat Monga, Samy Bengio, and Rafal Józefowicz. 2016. Revisiting distributed synchronous sgd. *Arxiv*, 1604.00981.

Mia Xu Chen, Orhan Firat, Ankur Bapna, Melvin Johnson, Wolfgang Macherey, George Foster, Llion Jones, Niki Parmar, Mike Schuster, Zhifeng Chen, Yonghui Wu, and Macduff Hughes. 2018. The best of both worlds: Combining recent advances in neural machine translation. *arxiv*, 1804.09849.

Adam Coates, Brody Huval, Tao Wang, David J. Wu, Bryan Catanzaro, and Andrew Y. Ng. 2013. Deep learning with cots hpc systems. In *Proc. of ICML*.

Matthieu Courbariaux, Yoshua Bengio, and Jean-Pierre David. 2015. Training deep neural networks with low precision multiplications.

Jeff Dean. 2017. Machine learning for systems and systems for machine learning. In *Proc. of NIPS Workshop on ML Systems*.

Jeffrey Dean, Gregory S. Corrado, Rajat Monga, Kai Chen, Matthieu Devin, Quoc V. Le, Mark Z. Mao, Marc'Aurelio Ranzato, Andrew W. Senior, Paul A. Tucker, Ke Yang, and Andrew Y. Ng. 2012. Large scale distributed deep networks. In *Proc. of NIPS*.

Ondrej Dusek and Filip Jurcícek. 2016. Sequence-to-sequence generation for spoken dialogue via deep syntax trees and strings. In *Proc. of ACL*.

Sergey Edunov, Myle Ott, and Sam Gross. 2017. Fairseq. `https://github.com/pytorch/fairseq`.

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

Priya Goyal, Piotr Dollár, Ross B. Girshick, Pieter Noordhuis, Lukasz Wesolowski, Aapo Kyrola, Andrew Tulloch, Yangqing Jia, and Kaiming He. 2017. Accurate, Large Minibatch SGD: Training ImageNet in 1 Hour. In *Proc. of CVPR*.

Hieu Hoang, Philipp Koehn, Ulrich Germann, Kenneth Heafield, and Barry Haddow. 2006. multi-bleu.perl. `https://github.com/moses-smt/mosesdecoder/blob/master/scripts/generic/multi-bleu.perl`.

Elad Hoffer, Itay Hubara, and Daniel Soudry. 2017. Train longer, generalize better: closing the generalization gap in large batch training of neural networks. In *Proc. of NIPS*, pages 1729–1739.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A Method for Stochastic Optimization. In *Proc. of ICLR*.

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondrej Bojar, Alexandra

Constantin, and Evan Herbst. 2007. Moses: Open source toolkit for statistical machine translation. In *ACL Demo Session*.

Jaroslaw Kutylowski. 2018. Deepl press information. https://www.deepl.com/press.html.

Sharan Narang, Gregory Diamos, Erich Elsen, Paulius Micikevicius, Jonah Alben, David Garcia, Boris Ginsburg, Michael Houston, Oleksii Kuchaiev, Ganesh Venkatesh, and Hao Wu. 2018. Mixed Precision Training. In *Proc. of ICLR*.

Myle Ott, Michael Auli, David Grangier, and MarcAurelio Ranzato. 2018. Analyzing uncertainty in neural machine translation. In *International Conference on Machine Learning (ICML)*.

ParaCrawl. 2018. ParaCrawl. http://paracrawl.eu/download.html.

Romain Paulus, Caiming Xiong, and Richard Socher. 2018. A deep reinforced model for abstractive summarization. In *Proc. of ICLR*.

Gabriel Pereyra, George Tucker, Jan Chorowski, Lukasz Kaiser, and Geoffrey E. Hinton. 2017. Regularizing neural networks by penalizing confident output distributions. In *Proc. of ICLR Workshop*.

Matt Post. 2018. A call for clarity in reporting bleu scores. *arXiv*, 1804.08771.

Christopher De Sa, Megan Leszczynski, Jian Zhang, Alana Marzoev, Christopher R. Aberger, Kunle Olukotun, and Christopher Ré. 2018. High-accuracy low-precision training. *Arxiv*, 1803.03383.

Abigail See, Peter J Liu, and Christopher D Manning. 2017. Get to the point: Summarization with pointer-generator networks. In *Proc. of ACL*.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Neural machine translation of rare words with subword units. In *Proc. of ACL*.

Iulian Serban, Alessandro Sordoni, Ryan Joseph Lowe, Laurent Charlin, Joelle Pineau, Aaron C. Courville, and Yoshua Bengio. 2017. A hierarchical latent variable encoder-decoder model for generating dialogues. In *Proc. of AAAI*.

Peter Shaw, Jakob Uszkoreit, and Ashish Vaswani. 2018. Self-attention with relative position representations. In *Proc. of NAACL*.

Patrice Y. Simard and Hans Peter Graf. 1993. Backpropagation without multiplication. In *Proc. of NIPS*.

Samuel L. Smith, Pieter-Jan Kindermans, and Quoc V. Le. 2018. Don't decay the learning rate, increase the batch size. In *Proc. of ICLR*.

Alessandro Sordoni, Michel Galley2, Michael Auli, Chris Brockett, Yangfeng Ji, Margaret Mitchell, Jian-Yun Nie, Jianfeng Gao, and Bill Dolan. 2015. A neural network approach to context-sensitive generation of conversational responses. In *Proc. of ACL*.

Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jonathon Shlens, and Zbigniew Wojna. 2015. Rethinking the Inception Architecture for Computer Vision. *arXiv preprint arXiv:1512.00567*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention Is All You Need. In *Proc. of NIPS*.

Hainan Xu and Philipp Koehn. 2017. Zipporah: a fast and scalable data cleaning system for noisy web-crawled parallel corpora. In *Proc. of EMNLP*.