FAIRSEQ: A Fast, Extensible Toolkit for Sequence Modeling

Myle Ott[△*] **Sergey Edunov**[△*] **Alexei Baevski**[△] **Angela Fan**[△] **Sam Gross**[△]
Nathan Ng[△] **David Grangier**[▽†] **Michael Auli**[△]
[△] Facebook AI Research
[▽] Google Brain

Abstract

FAIRSEQ is an open-source sequence modeling toolkit that allows researchers and developers to train custom models for translation, summarization, language modeling, and other text generation tasks. The toolkit is based on PyTorch and supports distributed training across multiple GPUs and machines. We also support fast mixed-precision training and inference on modern GPUs. A demo video can be found here: https://www.youtube.com/watch?v=OtgDdWtHvto.

1 Introduction

Neural sequence-to-sequence models have been successful on a variety of text generation tasks, including machine translation, abstractive document summarization, and language modeling. Accordingly, both researchers and industry professionals can benefit from a fast and easily extensible sequence modeling toolkit.

There are several toolkits with similar basic functionality, but they differ in focus area and intended audiences. For example, OpenNMT (Klein et al., 2017) is a community-built toolkit written in multiple languages with an emphasis on extensibility. MarianNMT (Junczys-Dowmunt et al., 2018) focuses on performance and the backend is written in C++ for fast automatic differentiation. OpenSeq2Seq (Kuchaiev et al., 2018) provides reference implementations for fast distributed and mixed precision training. Tensor2tensor (Vaswani et al., 2018) and Sockeye (Hieber et al., 2018) focus on production-readiness.

In this paper, we present FAIRSEQ, a sequence modeling toolkit written in PyTorch that is fast, extensible, and useful for both research and production. FAIRSEQ features: (i) a common interface across models and tasks that can be extended with user-supplied plug-ins (§2); (ii) efficient distributed and mixed precision training, enabling training over datasets with hundreds of millions of sentences on current hardware (§3); (iii) state-of-the-art implementations and pretrained models for machine translation, summarization, and language modeling (§4); and (iv) optimized inference with multiple supported search algorithms, including beam search, diverse beam search (Vijayakumar et al., 2016), and top-k sampling. FAIRSEQ is distributed with a BSD license and is available on GitHub at https://github.com/pytorch/fairseq.

[*] equal contribution
[†] Work done while at Facebook AI Research.

2 Design

Extensibility. FAIRSEQ can be extended through five types of user-supplied plug-ins, which enable experimenting with new ideas while reusing existing components as much as possible.

Models define the neural network architecture and encapsulate all learnable parameters. Models extend the `BaseFairseqModel` class, which in turn extends `torch.nn.Module`. Thus any FAIRSEQ model can be used as a stand-alone module in other PyTorch code. Models can additionally predefine named *architectures* with common network configurations (e.g., embedding dimension, number of layers, etc.). We also abstracted the methods through which the model interacts with the generation algorithm, e.g., beam search, through step-wise prediction. This isolates model implementation from the generation algorithm.

Criterions compute the loss given the model and a batch of data, roughly: `loss = criterion(model, batch)`. This formulation makes criterions very expressive, since they have complete access to the model. For example, a criterion may perform on-the-fly genera-

tion to support sequence-level training (Edunov et al., 2018b) or online backtranslation (Edunov et al., 2018a; Lample et al., 2018). Alternatively, in a mixture-of-experts model, a criterion may implement EM-style training and backpropagate only through the expert that produces the lowest loss (Shen et al., 2019).

Tasks store dictionaries, provide helpers for loading and batching data and define the training loop. They are intended to be immutable and primarily interface between the various components. We provide tasks for translation, language modeling, and classification.

Optimizers update the model parameters based on the gradients. We provide wrappers around most PyTorch optimizers and an implementation of Adafactor (Shazeer and Stern, 2018), which is a memory-efficient variant of Adam.

Learning Rate Schedulers update the learning rate over the course of training. We provide several popular schedulers, e.g., the inverse square-root scheduler from Vaswani et al. (2017) and cyclical schedulers based on warm restarts (Loshchilov and Hutter, 2016).

Reproducibility and forward compatibility. FAIRSEQ includes features designed to improve reproducibility and forward compatibility. For example, checkpoints contain the full state of the model, optimizer and dataloader, so that results are reproducible if training is interrupted and resumed. FAIRSEQ also provides forward compatibility, i.e., models trained using old versions of the toolkit will continue to run on the latest version through automatic checkpoint upgrading.

3 Implementation

FAIRSEQ is implemented in PyTorch and it provides efficient batching, mixed precision training, multi-GPU as well as multi-machine training.

Batching. There are multiple strategies to batch input and output sequence pairs (Morishita et al., 2017). FAIRSEQ minimizes padding within a mini-batch by grouping source and target sequences of similar length. The content of each mini-batch stays the same throughout training, however mini-batches themselves are shuffled randomly every epoch. When training on more than one GPU or machine, then the mini-batches for each worker

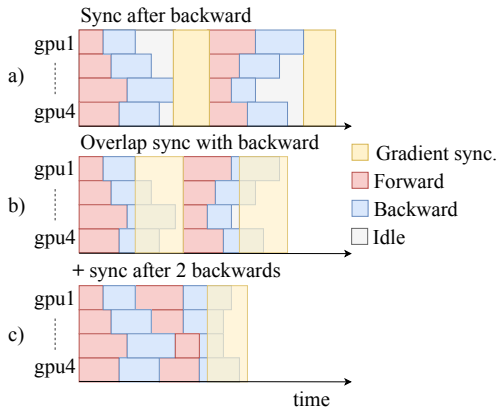

Figure 1: Illustration of (a) gradient synchronization and idle time during training, (b) overlapping back-propagation (backward) with gradient synchronization to improve training speed, (c) how accumulating gradient updates can reduce variance in processing time and reduce communication time.

are likely to differ in the average sentence length which results in more representative updates.

Multi-GPU training. FAIRSEQ uses the NCCL2 library and `torch.distributed` for inter-GPU communication. Models are trained in a synchronous optimization setup where each GPU has a copy of the model to process a sub-batch of data after which gradients are synchronized between GPUs; all sub-batches constitute a mini-batch. Even though sub-batches contain a similar number of tokens, we still observe a high variance in processing times. In multi-GPU or multi-machine setups, this results in idle time for most GPUs while slower workers are finishing their work (Figure 1 (a)). FAIRSEQ mitigates the effect of stragglers by overlapping gradient synchronization between workers with the backward pass and by accumulating gradients over multiple mini-batches for each GPU (Ott et al., 2018b).

Overlapping gradient synchronization starts to synchronize gradients of parts of the network when they are computed. In particular, when the gradient computation for a layer finishes, FAIRSEQ adds the result to a buffer. When the size of the buffer reaches a predefined threshold, the gradients are synchronized in a background thread while back-propagation continues as usual (Figure 1 (b)). Next, we accumulate gradients for multiple sub-batches on each GPU which reduces the variance in processing time between workers since there is no need to wait for stragglers after each sub-batch (Figure 1 (c)). This also increases the

	Sentences/sec
FAIRSEQ FP32	88.1
FAIRSEQ FP16	**136.0**

Table 1: Translation speed measured on a V100 GPU on the test set of the standard WMT'14 English-German benchmark using a big Transformer model.

effective batch size but we found that models can still be trained effectively (Ott et al., 2018b).

Mixed precision. Recent GPUs enable efficient half precision floating point (FP16) computation. FAIRSEQ provides support for both full precision (FP32) and FP16 at training and inference. We perform all forward-backward computations as well as the all-reduce for gradient synchronization between workers in FP16. However, the parameter updates remain in FP32 to preserve accuracy. FAIRSEQ implements dynamic loss scaling (Micikevicius et al., 2018) in order to avoid underflows for activations and gradients because of the limited precision offered by FP16. This scales the loss right after the forward pass to fit into the FP16 range while the backward pass is left unchanged. After the FP16 gradients are synchronized between workers, we convert them to FP32, restore the original scale, and update the weights.

Inference. FAIRSEQ provides fast inference for non-recurrent models (Gehring et al., 2017; Vaswani et al., 2017; Fan et al., 2018b; Wu et al., 2019) through incremental decoding, where the model states of previously generated tokens are cached in each active beam and re-used. This can speed up a naïve implementation without caching by up to an order of magnitude, since only new states are computed for each token. For some models, this requires a component-specific caching implementation, e.g., multi-head attention in the Transformer architecture.

During inference we build batches with a variable number of examples up to a user-specified number of tokens, similar to training. FAIRSEQ also supports inference in FP16 which increases decoding speed by 54% compared to FP32 with no loss in accuracy (Table 1).

4 Applications

FAIRSEQ has been used in many applications, such as machine translation (Gehring et al., 2017; Edunov et al., 2018b,a; Chen et al., 2018; Ott et al., 2018a; Song et al., 2018; Wu et al., 2019), language modeling (Dauphin et al., 2017; Baevski and Auli, 2019), abstractive document summarization (Fan et al., 2018a; Liu et al., 2018; Narayan et al., 2018), story generation (Fan et al., 2018b, 2019), error correction (Chollampatt and Ng, 2018), multilingual sentence embeddings (Artetxe and Schwenk, 2018), and dialogue (Miller et al., 2017; Dinan et al., 2019).

4.1 Machine translation

We provide reference implementations of several popular sequence-to-sequence models which can be used for machine translation, including LSTM (Luong et al., 2015), convolutional models (Gehring et al., 2017; Wu et al., 2019) and Transformer (Vaswani et al., 2017).

We evaluate a "big" Transformer encoder-decoder model on two language pairs, WMT English to German (En–De) and WMT 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 measure case-sensitive tokenized BLEU with multi-bleu (Hoang et al., 2006) and detokenized BLEU with SacreBLEU[1] (Post, 2018). All results use beam search with a beam width of 4 and length penalty of 0.6, following Vaswani et al. 2017. FAIRSEQ results are summarized in Table 2. We reported improved BLEU scores over Vaswani et al. (2017) by training with a bigger batch size and an increased learning rate (Ott et al., 2018b).

4.2 Language modeling

FAIRSEQ supports language modeling with gated convolutional models (Dauphin et al., 2017) and Transformer models (Vaswani et al., 2017). Models can be trained using a variety of input and output representations, such as standard token embeddings, convolutional character embeddings (Kim

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

	En–De	En–Fr
a. Gehring et al. (2017)	25.2	40.5
b. Vaswani et al. (2017)	28.4	41.0
c. Ahmed et al. (2017)	28.9	41.4
d. Shaw et al. (2018)	29.2	41.5
FAIRSEQ Transformer base	28.1	41.1
FAIRSEQ Transformer big	**29.3**	**43.2**
detok. SacreBLEU	*28.6*	*41.4*
8 GPU training time	*~12 h*	*~73 h*
128 GPU training time	*~1.3 h*	*~7.2 h*

Table 2: BLEU on news2014 for WMT English-German (En–De) and English-French (En–Fr). All results are based on WMT'14 training data, except for En–De (b), (c), (d) and our models which were trained on WMT'16. Train times based on V100 GPUs.

	Perplexity
Grave et al. (2016)	40.8
Dauphin et al. (2017)	37.2
Merity et al. (2018)	33.0
Rae et al. (2018)	29.2
FAIRSEQ Adaptive inputs	**18.7**

Table 3: Test perplexity on WikiText-103 (cf. Table 4).

et al., 2016), adaptive softmax (Grave et al., 2017), and adaptive inputs (Baevski and Auli, 2019). We also provide tutorials and pre-trained models that replicate the results of Dauphin et al. (2017) and Baevski and Auli (2019) on WikiText-103 and the One Billion Word datasets.

We evaluate two Transformer language models, which use only a decoder network and adaptive input embeddings, following Baevski and Auli (2019). The first model has 16 blocks, inner dimension 4K and embedding dimension 1K; results on WikiText-103 are in Table 3. The second model has 24 blocks, inner dimension 8K and embedding dimension 1.5K; results on the One Billion Word benchmark are in Table 4.

4.3 Abstractive document summarization

Next, we experiment with abstractive document summarization where we use a base Transformer to encode the input document and then generate a summary with a decoder network. We use the CNN-Dailymail dataset (Hermann et al., 2015; Nallapati et al., 2016) of news articles paired with multi-sentence summaries. We evaluate on

	Perplexity
Dauphin et al. (2017)	31.9
Józefowicz et al. (2016)	30.0
Shazeer et al. (2017)	28.0
FAIRSEQ Adaptive inputs	**23.0**

Table 4: Test perplexity on the One Billion Word benchmark. Adaptive inputs share parameters with an adaptive softmax.

	ROUGE		
	1	2	L
See et al. (2017)	39.5	17.3	36.4
Gehrmann et al. (2018)	41.2	18.7	38.3
FAIRSEQ	40.1	17.6	36.8
+ pre-trained LM	**41.6**	**18.9**	**38.5**

Table 5: Abstractive summarization results on the full-text version of CNN-DailyMail dataset.

the full-text version with no entity anonymization (See et al., 2017); we truncate articles to 400 tokens (See et al., 2017). We use BPE with 30K operations to form our vocabulary following Fan et al. (2018a). To evaluate, we use the standard ROUGE metric (Lin, 2004) and report ROUGE-1, ROUGE-2, and ROUGE-L. To generate summaries, we follow standard practice in tuning the minimum output length and disallow repeating the same trigram (Paulus et al., 2017). Table 5 shows results of FAIRSEQ. We also consider a configuration where we input pre-trained language model representations to the encoder network and this language model was trained on newscrawl and CNN-Dailymail, totalling 193M sentences.

5 Conclusion

We presented FAIRSEQ, a fast, extensible toolkit for sequence modeling that is scalable and suitable for many applications. In the future, we will continue the development of the toolkit to enable further research advances.

Acknowledgements

We thank Jonas Gehring for writing the original Lua/Torch version of fairseq.

References

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

Mikel Artetxe and Holger Schwenk. 2018. Massively multilingual sentence embeddings for zero-shot cross-lingual transfer and beyond. *arXiv*, abs/1812.10464.

Alexei Baevski and Michael Auli. 2019. Adaptive input representations for neural language modeling. In *Proc. of ICLR*.

Yun Chen, Victor OK Li, Kyunghyun Cho, and Samuel R Bowman. 2018. A stable and effective learning strategy for trainable greedy decoding. *arXiv*, abs/1804.07915.

Shamil Chollampatt and Hwee Tou Ng. 2018. A multilayer convolutional encoder-decoder neural network for grammatical error correction. *arXiv*, abs/1801.08831.

Yann N. Dauphin, Angela Fan, Michael Auli, and David Grangier. 2017. Language modeling with gated convolutional networks. In *Proc. of ICML*.

Emily Dinan, Stephen Roller, Kurt Shuster, Angela Fan, Michael Auli, and Jason Weston. 2019. Wizard of Wikipedia: Knowledge-powered conversational agents. In *Proc. of ICLR*.

Sergey Edunov, Myle Ott, Michael Auli, and David Grangier. 2018a. Understanding back-translation at scale. In *Conference of the Association for Computational Linguistics (ACL)*.

Sergey Edunov, Myle Ott, Michael Auli, David Grangier, et al. 2018b. Classical structured prediction losses for sequence to sequence learning. In *Proc. of NAACL*.

Angela Fan, David Grangier, and Michael Auli. 2018a. Controllable abstractive summarization. In *ACL Workshop on Neural Machine Translation and Generation*.

Angela Fan, Mike Lewis, and Yann Dauphin. 2018b. Hierarchical neural story generation. In *Proc. of ACL*.

Angela Fan, Mike Lewis, and Yann Dauphin. 2019. Strategies for structuring story generation. *arXiv*, abs/1902.01109.

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

Sebastian Gehrmann, Yuntian Deng, and Alexander M Rush. 2018. Bottom-up abstractive summarization. *arXiv*, abs/1808.10792.

Edouard Grave, Armand Joulin, Moustapha Cissé, David Grangier, and Hervé Jégou. 2017. Efficient softmax approximation for gpus. In *Proc. of ICML*.

Edouard Grave, Armand Joulin, and Nicolas Usunier. 2016. Improving neural language models with a continuous cache. *arXiv*, abs/1612.04426.

Karl Moritz Hermann, Tomas Kocisky, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. 2015. Teaching machines to read and comprehend. In *NIPS*.

Felix Hieber, Tobias Domhan, Michael Denkowski, David Vilar, Artem Sokolov, Ann Clifton, and Matt Post. 2018. Sockeye: A Toolkit for Neural Machine Translation. *arXiv*, abs/1712.05690.

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`.

Rafal Józefowicz, Oriol Vinyals, Mike Schuster, Noam Shazeer, and Yonghui Wu. 2016. Exploring the limits of language modeling. *arXiv*, abs/1602.02410.

Marcin Junczys-Dowmunt, Roman Grundkiewicz, Tomasz Dwojak, Hieu Hoang, Kenneth Heafield, Tom Neckermann, Frank Seide, Ulrich Germann, Alham Fikri Aji, Nikolay Bogoychev, André F. T. Martins, and Alexandra Birch. 2018. Marian: Fast neural machine translation in C++. In *Proc. of ACL 2018, System Demonstrations*.

Yoon Kim, Yacine Jernite, David Sontag, and Alexander M Rush. 2016. Character-aware neural language models. In *Proc. of AAAI*.

Guillaume Klein, Yoon Kim, Yuntian Deng, Jean Senellart, and Alexander M. Rush. 2017. OpenNMT: Open-source toolkit for neural machine translation. In *Proc. ACL*.

Oleksii Kuchaiev, Boris Ginsburg, Igor Gitman, Vitaly Lavrukhin, Carl Case, and Paulius Micikevicius. 2018. OpenSeq2Seq: Extensible Toolkit for Distributed and Mixed Precision Training of Sequence-to-Sequence Models. In *Proc. of Workshop for NLP Open Source Software*.

Guillaume Lample, Myle Ott, Alexis Conneau, Ludovic Denoyer, and Marc'Aurelio Ranzato. 2018. Phrase-based & neural unsupervised machine translation. In *Proc. of EMNLP*.

Chin-Yew Lin. 2004. Rouge: a package for automatic evaluation of summaries. In *ACL Workshop on Text Summarization Branches Out*.

Yizhu Liu, Zhiyi Luo, and Kenny Zhu. 2018. Controlling length in abstractive summarization using a convolutional neural network. In *Proc. of EMNLP*.

Ilya Loshchilov and Frank Hutter. 2016. Sgdr: Stochastic gradient descent with warm restarts. In *Proc. of ICLR*.

Minh-Thang Luong, Hieu Pham, and Christopher D Manning. 2015. Effective approaches to attention-based neural machine translation. In *Proc. of EMNLP*.

Stephen Merity, Nitish Shirish Keskar, and Richard Socher. 2018. An analysis of neural language modeling at multiple scales. *arXiv*, abs/1803.08240.

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

A. H. Miller, W. Feng, A. Fisch, J. Lu, D. Batra, A. Bordes, D. Parikh, and J. Weston. 2017. Parlai: A dialog research software platform. *arXiv*, abs/1705.06476.

Makoto Morishita, Yusuke Oda, Graham Neubig, Koichiro Yoshino, Katsuhito Sudoh, and Satoshi Nakamura. 2017. An empirical study of mini-batch creation strategies for neural machine translation. In *Proc. of WMT*.

Ramesh Nallapati, Bowen Zhou, Cicero dos Santos, Caglar Gulcehre, and Bing Xiang. 2016. Abstractive text summarization using sequence-to-sequence rnns and beyond. In *SIGNLL Conference on Computational Natural Language Learning*.

Shashi Narayan, Shay B Cohen, and Mirella Lapata. 2018. Don't give me the details, just the summary! topic-aware convolutional neural networks for extreme summarization. *arXiv*, abs/1808.08745.

Myle Ott, Michael Auli, David Grangier, and MarcAurelio Ranzato. 2018a. Analyzing uncertainty in neural machine translation. In *Proc. of ICML*.

Myle Ott, Sergey Edunov, David Grangier, and Michael Auli. 2018b. Scaling neural machine translation. In *Proc. of WMT*.

Romain Paulus, Caiming Xiong, and Richard Socher. 2017. A deep reinforced model for abstractive summarization. *arXiv preprint arXiv:1705.04304*.

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

Jack W. Rae, Chris Dyer, Peter Dayan, and Timothy P. Lillicrap. 2018. Fast parametric learning with activation memorization. *arXiv*, abs/1803.10049.

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

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

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

Noam Shazeer, Azalia Mirhoseini, Krzysztof Maziarz, Andy Davis, Quoc V. Le, Geoffrey E. Hinton, and Jeff Dean. 2017. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. *arXiv*, abs/1701.06538.

Noam Shazeer and Mitchell Stern. 2018. Adafactor: Adaptive learning rates with sublinear memory cost. *arXiv preprint arXiv:1804.04235*.

Tianxiao Shen, Myle Ott, Michael Auli, and Marc'Aurelio Ranzato. 2019. Mixture models for diverse machine translation: Tricks of the trade. *arXiv*, abs/1902.07816.

Kaitao Song, Xu Tan, Di He, Jianfeng Lu, Tao Qin, and Tie-Yan Liu. 2018. Double path networks for sequence to sequence learning. *arXiv*, abs/1806.04856.

A. Vaswani, S. Bengio, E. Brevdo, F. Chollet, A. N. Gomez, S. Gouws, L. Jones, Ł. Kaiser, N. Kalchbrenner, N. Parmar, R. Sepassi, N. Shazeer, and J. Uszkoreit. 2018. Tensor2Tensor for Neural Machine Translation. *arXiv*, abs/1803.07416.

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*.

Ashwin K Vijayakumar, Michael Cogswell, Ramprasath R Selvaraju, Qing Sun, Stefan Lee, David Crandall, and Dhruv Batra. 2016. Diverse beam search: Decoding diverse solutions from neural sequence models. *arXiv preprint arXiv:1610.02424*.

Felix Wu, Angela Fan, Alexei Baevski, Yann N. Dauphin, and Michael Auli. 2019. Pay less attention with lightweight and dynamic convlutions. In *Proc. of ICLR*.