The OpenNMT Neural Machine Translation Toolkit: 2020 Edition

Guillaume Klein guillaume.klein@systrangroup.com
François Hernandez fhernandez@ubiqus.com
Vincent Nguyen vnguyen@ubiqus.com
Jean Senellart jean.senellart@systrangroup.com

Abstract

OpenNMT is a multi-year open-source ecosystem for neural machine translation (NMT) and natural language generation (NLG). The toolkit consists of multiple projects to cover the complete machine learning workflow: from data preparation to inference acceleration. The systems prioritize efficiency, modularity, and extensibility with the goal of supporting research into model architectures, feature representations, and source modalities, while maintaining API stability and competitive performance for production usages. OpenNMT has been used in several production MT systems and cited in more than 700 research papers.

1 Introduction

Neural machine translation (NMT) is a recent approach for machine translation that has led to remarkable improvements, particularly in terms of human evaluation, compared to rule-based and statistical machine translation (SMT) systems (Wu et al., 2016). From the initial work on recurrent sequence-to-sequence models (Sutskever et al., 2014) to recent advances on self-attentional models (Vaswani et al., 2017), a significant amount of contributions explored various model architectures, hyper-parameter settings, and data preparation techniques. This exploration is often constrained by engineering issues related to computation efficiency, code design, and model deployment.

In this context, we introduced OpenNMT [1] in late 2016, the first large audience open-source toolkit to design, train, and deploy neural machine translation models. The OpenNMT initiative consists of several projects to assist researchers and developers in their NMT journey, from data preparation to inference acceleration. It supports a wide range of model architectures and training procedures for neural machine translation as well as related tasks such as natural language generation and language modeling.

The open source community around neural machine translation is very active and includes several other projects that have similar goals and capabilities such as Fairseq (Ott et al., 2019), Sockeye (Hieber et al., 2017), or Marian (Junczys-Dowmunt et al., 2018). In the ongoing development of OpenNMT, we aim to build upon the strengths of those systems, while providing unique features and technology support.

In this paper, we briefly give an overview of the OpenNMT project and its history. We then present the key features that make OpenNMT particularly suited for research and production. Finally, we share some benchmarks for comparison and current work directions.

[1] `https://opennmt.net`

2 Project overview

OpenNMT is a collection of projects supporting easy adoption of neural machine translation and related tasks. The project has two actively maintained implementations:

- **OpenNMT-py**: A user-friendly and multimodal implementation benefiting from PyTorch ease of use and versatility.
- **OpenNMT-tf**: A modular and stable implementation powered by the TensorFlow 2 ecosystem.

These implementations provide command line utilities and a Python library to configure, train, and run models. Each implementation has its own design and set of features, but both share the goals that started the OpenNMT initiative: ease of use, efficiency, modularity, extensibility, and production readiness. The supported features are compared in Table 1.

Training	**py**	**tf**
Automatic evaluation	✓	✓
Checkpoint averaging	✓	✓
Contrastive learning		✓
Early stopping	✓	✓
Gradient accumulation	✓	✓
Supervised alignment	✓	✓
Mixed precision	✓	✓
Moving average	✓	✓
Multi-GPU	✓	✓
Multi-node	✓	✓
Online tokenization		✓
Pretrained embeddings	✓	✓
Scheduled sampling		✓
Vocabulary update		✓
Weighted dataset	✓	✓

Tasks	**py**	**tf**
Image/Video to text	✓	
Language modeling		✓
Sequence classification		✓
Sequence tagging		✓
Sequence to sequence	✓	✓
Speech to text	✓	✓
Summarization	✓	

Models	**py**	**tf**
ConvS2S	✓	
DeepSpeech2	✓	
GPT-2		✓
Im2Text	✓	
Listen, Attend and Spell		✓
RNMT+		✓
RNN with attention	✓	✓
Transformer	✓	✓

Decoding	**py**	**tf**
Beam search	✓	✓
Coverage penalty	✓	✓
CTranslate2 compatibility	✓	✓
Ensemble	✓	
Length penalty	✓	✓
N-best rescoring	✓	✓
N-gram blocking	✓	
Phrase table	✓	
Random noise	✓	✓
Random sampling	✓	✓
Replace unknown	✓	✓

Model configuration	**py**	**tf**
Copy attention	✓	
Coverage attention	✓	
Hybrid models		✓
Multi source		✓
Multiple input features	✓	✓
Relative position	✓	✓
Tied embeddings	✓	✓

Table 1: Features implemented by OpenNMT-py (column **py**) and OpenNMT-tf (column **tf**).

The ecosystem is completed by tools that are used in both OpenNMT-py and OpenNMT-tf:

- **Tokenizer**: A fast and customizable text tokenization library that includes Unicode-based segmentation, subword training and encoding (BPE and SentencePiece), protected sequences, and case annotation.
- **CTranslate2**: An optimized inference engine for Transformer models which comes with a custom C++ implementation supporting fast CPU and GPU execution, quantization, parallel translations, and interactive decoding.

Finally, these different projects are integrated in **nmt-wizard-docker** which proposes a way to containerize NMT frameworks and provide a unique command line interface for training, translating, and serving models.

All projects are released on GitHub[2] under the MIT license.

[2] `https://github.com/OpenNMT`

2.1 History

OpenNMT was first released in late 2016 as a Torch7 implementation. This version was the result of a collaboration between Harvard NLP and SYSTRAN and was based on seq2seq-attn, an open-source project developed by Harvard student Yoon Kim. The original demonstration paper (Klein et al., 2017) was awarded "Best Demonstration Paper Runner-Up" at ACL 2017.

After the release of PyTorch (Paszke et al., 2019), the Facebook A.I. Research team shared a complete rewrite of the project that later became OpenNMT-py and initiated the sunsetting of the Torch7 version of OpenNMT. The ecosystem was then extended with OpenNMT-tf which prioritized production, while OpenNMT-py had a focus on research at this time.

After more than 3 years of active development, OpenNMT projects have been starred by over 7,400 users. A community forum[3] is also home of 970 users and more than 9,800 posts about NMT research and how to use OpenNMT effectively.

2.2 Adoption

Cited in over 700 scientific publications as of May 2020, OpenNMT has also been directly used in numerous research papers. The system is employed both to conduct new experiments and as a baseline for sequence-to-sequence approaches. OpenNMT was used for other tasks related to neural machine translation such as summarization (Gehrmann et al., 2018), data-to-text (Wiseman et al., 2017), image-to-text (Deng et al., 2017), automatic speech recognition (Ericson, 2019) and semantic parsing (van Noord and Bos, 2017).

OpenNMT also proved to be widespread in industry. Companies such as SYSTRAN (Crego et al., 2016), Booking.com (Levin et al., 2017), or Ubiqus[4] are known to deploy OpenNMT models in production. We note that a number of industrial entities published scientific papers showing their internal experiments using the framework such as SwissPost (Girletti et al., 2019) and BNP Paribas (Mghabbar and Ratnamogan, 2020), while NVIDIA used OpenNMT as a benchmark for the release of TensorRT 6[5].

3 Key features

3.1 Model architectures catalog

While OpenNMT initially focused on sequence-to-sequence models applied to translation, it has been extended to support additional architectures and model components (see Table 1).

Multiple architectures. The toolkits implement the most used architectures for neural machine translation: recurrent with attention (Bahdanau et al., 2015; Luong et al., 2015), self-attentional (Vaswani et al., 2017), and convolutional (Gehring et al., 2017). The project also includes components for other tasks such as encoders for non-text inputs, decoders for generative languages models, and copy attention mechanisms (See et al., 2017) for summarization.

Modular design. We focus on modularity to allow ideas from one paper to be reused in another context. OpenNMT-tf pushes this mindset to the extreme by requiring users to configure their model via Python code. This enables a high level of modelling freedom to support custom architectures such as hybrid sequence-to-sequence models (Chen et al., 2018), multi-source Transformer models (Libovický et al., 2018), and nested input features.

[3] https://forum.opennmt.net/

[4] https://slator.com/features/how-ubiqus-deploys-neural-machine-translation-in-language-operations/

[5] https://news.developer.nvidia.com/tensorrt6-breaks-bert-record

	Model size	CPU	GTX1080	GTX1080Ti	RTX2080Ti	BLEU
OpenNMT-tf	367MB	217.6	1659.2	1762.8	1628.3	26.9
OpenNMT-py	542MB	179.1	1510.0	1709.3	1406.2	26.7
CTranslate2	374MB	389.4	3081.3	3388.0	4196.2	26.7
+ *int16*	197MB	413.6	3055.7	3380.4	4202.9	26.7
+ *int8*	110MB	508.3	2654.8	2734.6	3143.4	26.8
+ *vmap*	121MB	646.2	2921.5	2992.1	3312.9	26.6

Table 2: This table compares model size and translation speed (target tokens per second) for a base English-German Transformer. The BLEU scores are computed on an undisclosed test set and are reported to show that quality is comparable. All runs use a batch size of 32 and a beam size of 4. The CTranslate2 models are generated from the OpenNMT-py model weights. There is a noticeable drop in performance on RTX for both -tf and -py for an unknown reason at the time of this publication.

3.2 Scalable training

As GPU performance has been drastically improving in the last few years, it has been important for software to take it into account not to introduce unnecessary bottlenecks. Both OpenNMT-py and OpenNMT-tf have been constantly optimized to keep up with the available computing power and its intricacies, as well as most mainstream training methods.

Optimized and parallel training. Training efficiency was an early focus of OpenNMT. It can run the training on multiple GPUs and machines using data parallelism. OpenNMT implementations are also compatible with mixed precision training to make use of NVIDIA's Tensor Cores, and when possible, they employ graph execution.

Optimizations for low-resource hardware. We also strive to make NMT training possible on low memory systems. OpenNMT supports gradient accumulation which is a way to simulate larger batch sizes that do not fit on the available GPU memory. OpenNMT-py is also able to shard the loss computation to reduce memory usage.

Efficient data pipeline. Both OpenNMT implementations make use of a producer-consumer design to feed examples to the training. This effectively reduces the data pipeline latency as well as the overall memory usage.

3.3 Optimized and interactive inference engine

CTranslate2 is a unique project that accelerates the execution of Transformer variants trained with OpenNMT-py and OpenNMT-tf. It is a custom C++ implementation with no runtime dependency on PyTorch or TensorFlow. The engine can run on CPU and GPU and is up to 4 times faster than a baseline PyTorch execution as shown in Table 2.

Specialized engine. As the engine focuses on executing specific model architectures, it implements graph-level optimizations such as layer fusion, memory reusing, and caching.

Fast backend. On CPU, most of the heavy lifting is done by Intel MKL which is a reference library when it comes to high performance matrix operations. On GPU, the implementation uses several libraries provided by NVIDIA: Thrust, cuBLAS, and TensorRT.

Model quantization. Quantization support in deep learning frameworks is often incomplete, especially for sequence models. However, CTranslate2 fully supports 16-bit and 8-bit GEneral Matrix Multiplication (GEMM) to reduce the memory and computation requirements.

Multi-level parallelism. The parallelism can be configured at the batch and file levels. The first level refers to the number of OpenMP threads used to execute a batch while the second

corresponds to the number of batches that are executed in parallel. This multi-level approach allows to trade-off latency and throughput.

Decoding optimizations. Multiple decoding tricks can be combined to accelerate speed, for instance restricting target vocabulary using a pretrained mapping, removing finished translations from the batch, sorting sentences by length, or skipping last Softmax in greedy decoding.

Interactive decoding. CTranslate2 also supports interactive decoding features such as auto-completing partial translations and returning alternatives at a specific position in the translation.

3.4 Efficient and customizable tokenization

Tokenizer is a standalone project that provides practical C++ and Python APIs for text tokenization. It focuses on efficiency and includes features that we found useful for training high-quality translation models.

Configurable reversibility. The tokenization can be made reversible by marking either joints or spaces. These markers can be attached to the input tokens or injected as separate tokens.

Advanced text segmentation. Several flags can finely control where to segment: on digits, on alphabet change, on case change, etc. Additionally, we introduced special control characters to prevent segmentation in delimited sequences.

Subword training and encoding. The project can train and apply SentencePiece (Kudo and Richardson, 2018) and BPE (Sennrich et al., 2016) models, while making them compatible with all features listed above.

3.5 Model serving

OpenNMT also provides components to serve translation models. OpenNMT-py includes a REST server that can manage multiple models and unload those that are not used. The server integrates CTranslate2 for efficient execution and Tokenizer for on-the-fly tokenization.

OpenNMT-tf models are compatible with TensorFlow Serving which is a scalable solution to serve machine learning models. OpenNMT-tf models are also compatible with the OpenNMT-py REST server via the CTranslate2 integration.

4 Benchmarks

OpenNMT, as a comprehensive project geared towards Neural Machine Translation, can be used in a fairly straightforward way to build SOTA systems and experiment on various tasks. Anyone can throw together a competitive system using the right data, processing and training procedures. For instance, the results for English to German translation in Table 3 are obtained with the following configuration:

- **Data**: English to German WMT19 task, with the addition of ParaCrawl v5 instead of v3.
- **Tokenization**: 40,000 BPE merge operations, learned and applied with Tokenizer.
- **Model**: Transformer Medium (12 $heads$, 768 d_{model} size, 3072 d_{ff} size).
- **Training**: Trained with OpenNMT-py on 6 RTX 2080 Ti, using mixed precision. Initial batch size is around 50,000 tokens, final batch size around 200,000 tokens.
- **Inference**: Shown scores are obtained with beam search of size 5 and average length penalty.

This vanilla system is constrained to the WMT rules to facilitate reproducibility. The commercial systems it is compared to are not constrained and we don't know the extent of the additional data that may be used. It is also very likely they use some bigger models, when we restrained this experiment to a Transformer Medium to keep the computation budget reasonable.

System	**nt14**	**nt15**	**nt16**	**nt17**	**nt18**	**nt19**	Wall time	GPU time
@30k steps	30.9	33.1	38.1	31.3	47.2	40.6	8h	48h
@100k steps	32.9	34.5	39.3	32.8	47.7	41.1	54h	324h
Online G	30.9	33.9	38.6	31.6	48.0	43.9	-	
Online M	32.3	34.3	40.5	33.1	48.8	43.8	-	

Table 3: OpenNMT system vs. some commercial systems. (BLEU scores obtained with SacreBLEU v1.3.7, online translations performed in April 2020.)

During the WMT19 campaign (Barrault et al., 2019), the best BLEU score for English to German was 44.9 but the best human evaluated system scored only 42.7 with an ensemble of Big Tranformers. It is also important to stress that we found many WMT19 references in the test sets (whether for English to German or some other pairs) were obviously being post edits of commercial systems. A very simple way to outline this phenomenon was to score each document with these commercial systems and show the huge difference in BLEU points for some of them. On the other hand, these systems were not over-performing in the same way for previous years test sets. This has been reported by several papers in the WMT19 campaign.

Extending this to a more complete setup, with internal datasets as well as a bigger architecture, OpenNMT tools allow to reach a superior performance. Some results for an internal English to French setup are presented in Table 4.

System	**nt14**	**finance**	**legal**	**general**	**life sciences**
OpenNMT	43.2	46.6	37.2	39.6	55.9
Online G	43.3	43.6	33.9	37.9	50.8
Online M	37.6	36.2	28.2	36.7	46.4

Table 4: OpenNMT English to French model performance on test sets of various domains. (BLEU scores obtained with SacreBLEU v1.3.7, online translations performed in April 2020.)

5 Current work

As OpenNMT aims to provide a state-of-the-art ecosystem for neural machine translation, we are continuously reproducing published papers with the goal of cherry-picking features with significant impact and implementing technologies that are part of the complete translation workflow. For example, we are currently working on combining translation memories with NMT.

We are also interested in further accelerating model inference via the CTranslate2 project. Future works include a better support of reduced precision on GPU and integration of optimized backends for non-Intel CPUs.

Finally, we are planning on massively releasing ready-to-use state-of-the-art models for a collection of language pairs to simplify the adoption of NMT.

6 Summary

We presented OpenNMT, an open-source ecosystem for neural machine translation and natural language generation. The toolkit contains multiple projects and features to cover the complete model production workflow. The main implementations, OpenNMT-py and OpenNMT-tf, support many configurable models and efficient training procedures to produce high-quality models. We also published CTranslate2, an optimized inference engine which to our knowledge is one of the fastest decoder of Transformer models. All projects are available on GitHub at `https://github.com/OpenNMT`.

References

Bahdanau, D., Cho, K., and Bengio, Y. (2015). Neural machine translation by jointly learning to align and translate. In Bengio, Y. and LeCun, Y., editors, *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

Barrault, L., Bojar, O., Costa-jussÃ, M. R., Federmann, C., Fishel, M., Graham, Y., Haddow, B., Huck, M., Koehn, P., Malmasi, S., Monz, C., MÃller, M., Pal, S., Post, M., and Zampieri, M. (2019). Findings of the 2019 conference on machine translation (wmt19). In *Proceedings of the Fourth Conference on Machine Translation (Volume 2: Shared Task Papers, Day 1)*, pages 1–61, Florence, Italy. Association for Computational Linguistics.

Chen, M. X., Firat, O., Bapna, A., Johnson, M., Macherey, W., Foster, G. F., Jones, L., Parmar, N., Schuster, M., Chen, Z., Wu, Y., and Hughes, M. (2018). The best of both worlds: Combining recent advances in neural machine translation. *CoRR*, abs/1804.09849.

Crego, J., Kim, J., Klein, G., Rebollo, A., Yang, K., Senellart, J., Akhanov, E., Brunelle, P., Coquard, A., Deng, Y., Enoue, S., Geiss, C., Johanson, J., Khalsa, A., Khiari, R., Ko, B., Kobus, C., Lorieux, J., Martins, L., Nguyen, D.-C., Priori, A., Riccardi, T., Segal, N., Servan, C., Tiquet, C., Wang, B., Yang, J., Zhang, D., Zhou, J., and Zoldan, P. (2016). Systran's pure neural machine translation systems.

Deng, Y., Kanervisto, A., Ling, J., and Rush, A. M. (2017). Image-to-markup generation with coarse-to-fine attention. In Precup, D. and Teh, Y. W., editors, *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017*, volume 70 of *Proceedings of Machine Learning Research*, pages 980–989. PMLR.

Ericson, L. (2019). Opensat19 pashto sad/asr/kws using opennmt and pyaudioanalysis.

Gehring, J., Auli, M., Grangier, D., Yarats, D., and Dauphin, Y. N. (2017). Convolutional sequence to sequence learning. *CoRR*, abs/1705.03122.

Gehrmann, S., Deng, Y., and Rush, A. (2018). Bottom-up abstractive summarization. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 4098–4109, Brussels, Belgium. Association for Computational Linguistics.

Girletti, S., Bouillon, P., Bellodi, M., and Ursprung, P. (2019). Preferences of end-users for raw and post-edited nmt in a business environment. In *Proceedings of the 41st Conference Translating and the Computer*, pages 47–59.

Hieber, F., Domhan, T., Denkowski, M., Vilar, D., Sokolov, A., Clifton, A., and Post, M. (2017). Sockeye: A toolkit for neural machine translation. *CoRR*, abs/1712.05690.

Junczys-Dowmunt, M., Grundkiewicz, R., Dwojak, T., Hoang, H., Heafield, K., Neckermann, T., Seide, F., Germann, U., Fikri Aji, A., Bogoychev, N., Martins, A. F. T., and Birch, A. (2018). Marian: Fast neural machine translation in C++. In *Proceedings of ACL 2018, System Demonstrations*, pages 116–121, Melbourne, Australia. Association for Computational Linguistics.

Klein, G., Kim, Y., Deng, Y., Senellart, J., and Rush, A. (2017). OpenNMT: Open-source toolkit for neural machine translation. In *Proceedings of ACL 2017, System Demonstrations*, pages 67–72, Vancouver, Canada. Association for Computational Linguistics.

Kudo, T. and Richardson, J. (2018). SentencePiece: A simple and language independent subword tokenizer and detokenizer for neural text processing. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 66–71, Brussels, Belgium. Association for Computational Linguistics.

Levin, P., Dhanuka, N., and Khalilov, M. (2017). Machine translation at booking.com: Journey and lessons learned.

Libovický, J., Helcl, J., and Mareček, D. (2018). Input combination strategies for multi-source transformer decoder. In *Proceedings of the Third Conference on Machine Translation: Research Papers*, pages 253–260, Brussels, Belgium. Association for Computational Linguistics.

Luong, T., Pham, H., and Manning, C. D. (2015). Effective approaches to attention-based neural machine translation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1412–1421, Lisbon, Portugal. Association for Computational Linguistics.

Mghabbar, I. and Ratnamogan, P. (2020). Building a multi-domain neural machine translation model using knowledge distillation.

Ott, M., Edunov, S., Baevski, A., Fan, A., Gross, S., Ng, N., Grangier, D., and Auli, M. (2019). fairseq: A fast, extensible toolkit for sequence modeling. In *Proceedings of NAACL-HLT 2019: Demonstrations*.

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Kopf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., and Chintala, S. (2019). Pytorch: An imperative style, high-performance deep learning library. In Wallach, H., Larochelle, H., Beygelzimer, A., dAlché-Buc, F., Fox, E., and Garnett, R., editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc.

See, A., Liu, P. J., and Manning, C. D. (2017). Get to the point: Summarization with pointer-generator networks. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1073–1083, Vancouver, Canada. Association for Computational Linguistics.

Sennrich, R., Haddow, B., and Birch, A. (2016). Neural machine translation of rare words with subword units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1715–1725, Berlin, Germany. Association for Computational Linguistics.

Sutskever, I., Vinyals, O., and Le, Q. V. (2014). Sequence to sequence learning with neural networks. In Ghahramani, Z., Welling, M., Cortes, C., Lawrence, N. D., and Weinberger, K. Q., editors, *Advances in Neural Information Processing Systems 27*, pages 3104–3112. Curran Associates, Inc.

van Noord, R. and Bos, J. (2017). Neural semantic parsing by character-based translation: Experiments with abstract meaning representations. *CoRR*, abs/1705.09980.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L. u., and Polosukhin, I. (2017). Attention is all you need. In Guyon, I., Luxburg, U. V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., and Garnett, R., editors, *Advances in Neural Information Processing Systems 30*, pages 5998–6008. Curran Associates, Inc.

Wiseman, S., Shieber, S., and Rush, A. (2017). Challenges in data-to-document generation. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2253–2263, Copenhagen, Denmark. Association for Computational Linguistics.

Wu, Y., Schuster, M., Chen, Z., Le, Q. V., Norouzi, M., Macherey, W., Krikun, M., Cao, Y., Gao, Q., Macherey, K., Klingner, J., Shah, A., Johnson, M., Liu, X., Łukasz Kaiser, Gouws, S., Kato, Y., Kudo, T., Kazawa, H., Stevens, K., Kurian, G., Patil, N., Wang, W., Young, C., Smith, J., Riesa, J., Rudnick, A., Vinyals, O., Corrado, G., Hughes, M., and Dean, J. (2016). Google's neural machine translation system: Bridging the gap between human and machine translation. *CoRR*, abs/1609.08144.