Neural Monkey: The Current State and Beyond

Jindřich Helcl
Jindřich Libovický
Tom Kocmi
Tomáš Musil
Ondřej Cífka
Dušan Variš
Ondřej Bojar [surname]@ufal.mff.cuni.cz
Institute of Formal and Applied Linguistics
Faculty of Mathematics and Physics, Charles University
Malostranské náměstí 25, Prague, 118 00, Czech Republic

Abstract

Neural Monkey is an open-source toolkit for sequence-to-sequence learning. The focus of this paper is to present the current state of the toolkit to the intended audience, which includes students and researchers, both active in the deep learning community and newcomers. For each of these target groups, we describe the most relevant features of the toolkit, including the simple configuration scheme, methods of model inspection that promote useful intuitions, or a modular design for easy prototyping. We summarize relevant contributions to the research community which were made using this toolkit and discuss the characteristics of our toolkit with respect to other existing systems. We conclude with a set of proposals for future development.

1 Introduction

Neural Monkey (Helcl and Libovický, 2017) is an open-source tool for sequential learning developed and maintained at Charles University. It is used mainly for experiments with neural machine translation (NMT), but also for other natural language processing (NLP) tasks which use deep learning, such as image captioning, multimodal translation, optical character recognition, text summarization or sentiment analysis. In this paper, we present the state of the software at the beginning of 2018 and share the plans for future development.

Neural Monkey is best suited for research and education. Our ambition is not developing a production-ready software. We also do not focus on technical tricks to achieve the best performance. Instead, we provide an extensible library of implemented features with references to the papers where they were proposed.

We try to align our aims with the needs of three possibly overlapping groups of users. First, we want to help students understand how various models work and build intuition for their training. Second, we want to provide a simple tool for newcomers to the field of deep learning to help them start experimenting with these models. Third, we design the toolkit in a highly modular fashion with replaceable components to suit researchers who want to try out their ideas without having to implement everything from scratch.

Neural Monkey is implemented in Python 3.5 using the TensorFlow library (Abadi et al., 2016). We share the source code on GitHub,[1] where we also report and solve issues, provide support for users and discuss the development, so everyone can contribute with their ideas.

This paper is organized as follows. Section 2 presents the goals of the development of the toolkit with respect to the needs of the considered user groups. Section 3 gives an overview of the toolkit features. In Section 4, we summarize the research contributions that have been made with Neural Monkey. We discuss the differences from other toolkits in Section 5. We propose our plans for the future development and conclude in Section 6.

2 The Goals of Neural Monkey Development

During the development of Neural Monkey, we keep in mind three groups of potential users. In this section, we discuss the interests of the members of each group, and establish a set of goals aligned with the considered interests.

The first interest group are researchers or NLP engineers who would like to start using deep learning. Neural Monkey is easy to install; it uses only a few Python modules which can be installed using *PiPy*, the Python package manager. The experiments are configured using configuration files with a simple *ini* syntax enriched with elements from Python. The sections of a configuration file correspond to an established conceptualization of the problem as used in the NLP community. We provide several examples of configuration files which can be used as starting points for experiments. A minimal working example configuration for NMT is shown in Example 1.

The second interest group we consider are students. Most of the modules that implement network components correspond to research papers which propose the ideas. The corresponding paper is cited in the code, often including references to a particular section or equation in the paper. We believe that reading such a commented code can help understand the relation between equations in papers and the actual implementation.

Finally, the third group of our potential users are researchers in NLP, who are already experienced in deep learning. For researchers, we provide a collection of modular prototypes and algorithmic features with a convenient experiment management system.

The four main goals of Neural Monkey development are:

Code Readability. The code is written with an emphasis on readability. This includes referencing the papers and original implementations. Research software often does not focus on the quality of the code. With Neural Monkey, we are trying to have a clear code base, such that everyone can easily fork the repository and build their experimental setup on top of that.

Modularity along Research Concepts. While working on neural models, it is common to reuse existing implementation of the state-of-the-art architectures or apply different training techniques. For example, one could only modify the encoder in order to improve the input sequence representation.

Up-to-Date Building Blocks. Deep learning is a rapidly developing field with new architectures appearing every few months. Neural Monkey is a maintained collection of these models, ready for use in NLP experiments.

Fast Prototyping. We focus on fast and easy prototyping of new architectures and testing existing architectures on new tasks. We try to keep a clear API that allows reusing architecture components between various tasks.

[1] https://github.com/ufal/neuralmonkey

```
[main]
output="output_dir"
batch_size=64
epochs=20
train_dataset=<train_data>
val_dataset=<val_data>
trainer=<trainer>
runners=[<runner>]
evaluation=[("target", evaluators.BLEU)]
logging_period=500
validation_period=5000

[train_data]
class=dataset.from_files
s_source="data/train.en"
s_target="data/train.de"
lazy=True

[val_data]
class=dataset.from_files
s_source="data/val.en"
s_target="data/val.de"

[encoder_vocabulary]
class=vocabulary.from_wordlist
path="en_vocab.tsv"

[decoder_vocabulary]
class=vocabulary.from_wordlist
path="de_vocab.tsv"

[encoder]
class=encoders.SentenceEncoder
rnn_size=500
embedding_size=600
data_id="source"
vocabulary=<encoder_vocabulary>

[attention]
class=attention.Attention
encoder=<encoder>

[decoder]
class=decoders.Decoder
encoders=[<encoder>]
attentions=[<attention>]
rnn_size=1000
embedding_size=600
data_id="target"
vocabulary=<decoder_vocabulary>

[trainer]
class=trainers.CrossEntropyTrainer
decoders=[<decoder>]
clip_norm=1.0

[runner]
class=runners.GreedyRunner
decoder=<decoder>
```

Example 1: A minimal example of a configuration file for an attentive RNN sequence-to-sequence machine translation model with the as presented by Bahdanau et al. (2014).

3 Toolkit Features

In the previous section, we defined our goals with respect to the needs of the target user groups. In this section, we provide a high-level overview of the features that are implemented in Neural Monkey. Further implementation details are explained in the following sections. In general, Neural Monkey distinguishes *encoders* (components that read data and produce a representation), and *decoders* (components that produce an output).

3.1 Processing Input ("Encoders")

This section summarizes the encoder components implemented in Neural Monkey. Each encoder processes an input into either a single state, a (temporal) sequence of states, or a two-dimensional (spatial) set of states.

- *Recurrent Sequence Encoder* allows encoding input sequence using stacked recurrent neural networks (RNNs). In addition to embedded symbolic input (Sutskever et al., 2014), we allow also a numerical input which can be used for instance for speech recognition.
- *RNN over Character-Level Convolutional Neural Network* (Lee et al., 2017), a technique that applies convolutional neural network (CNN) on the unsegmented character-level input. To produce the sequence of states, it uses a recurrent sequence encoder on the CNN outputs.
- *Deep Convolutional Encoder* (Gehring et al., 2017), an encoder for sequence-to-sequence learning with stacked convolutional layers and positional embeddings.

- *Transformer Encoder* is the encoder part of the Transformer (Vaswani et al., 2017) architecture which repeatedly uses the self-attention mechanism over the input representation.

- *ConvNet for Sentence Classification.* We implement the convolutional network with max-pooling for sentence classification (Kim, 2014). It is a fast baseline method for sentence classification such as sentiment analysis.

- *Self-Attentive Sentence Embedding.* As a more advanced technique for sentence classification, we use sentence encoding by Lin et al. (2017) which uses attention to produce a fixed-size structured representation of the input sequence.

- *Wrapper for ImageNet Networks.* Tasks combining language and vision usually use image features from deep convolutional networks for image classification. Neural Monkey provides wrappers for the inclusion of these models as implemented in TensorFlow Slim library.[2] In particular, we support VGG networks (Simonyan and Zisserman, 2014) and variants of ResNet (He et al., 2016).

- *Custom CNN with Residual Connections.* We also allow deep convolutional networks for image processing trained from scratch with an arbitrary architecture of stacked standard convolutional and max-pooling layers and residual blocks. (He et al., 2016)

3.2 Generating Output ("Decoders")

The list below describes components that generate an output, which we call decoders. Note that some of them have hidden states and if they implement the same API as the input processing model parts, they can be stacked with other decoders. Thanks to this feature, we can for instance apply a sequence labeler both on top of a recurrent encoder and a recurrent decoder and train them jointly.

- *Attentive Recurrent Sequence Decoder.* We implement the attention model introduced by Bahdanau et al. (2014) which was the main technique for a sequence decoding in the last few years. We support several recurrent unit implementations, including LSTM (Hochreiter and Schmidhuber, 1997), GRU (Cho et al., 2014), and conditional GRU (Firat and Cho, 2016). On top of the standard attention, we support the scaled dot-product attention (Vaswani et al., 2017), attention with the sentinel gate (Lu et al., 2016), and advanced attention combination strategies which explicitly model the attention over multiple sources (Libovický and Helcl, 2017).

- *Transformer Decoder*, the decoder part of the self-attentive sequence-to-sequence learning architecture mentioned above (Vaswani et al., 2017).

- *Sequence Labeler.* While sequence labeling, a multilayer perceptron is applied on the hidden states of the encoder. Additionally, the conditional random field (Lafferty et al., 2001; Do and Artieres, 2010) can be used to produce the output labeling.

- *Connectionist Temporal Classification*, a non-autoregressive technique of sequence decoding (Graves et al., 2006) used mostly in speech recognition and optical character recognition where the output labels keep the temporal order of the input signal.

- *Multilayer perceptron* for classification or regression.

[2] https://github.com/tensorflow/models/tree/master/research/slim

3.3 Additional Features

This section summarizes additional useful features available in Neural Monkey.

- *Beam Search and Model Ensembles.*
- *Independent Training, Saving and Loading Parts of the Model.*
- *Multi-Task Learning.* Experiment configurations can include definition of multiple decoders and specify which contribute to the training loss.
- *Loading Nematus Models.* Models that have been trained with Nematus (Sennrich et al., 2017b) can be loaded into Neural Monkey. This can be used either for fine-tuning, domain adaptation, or multi-task learning.
- *Bandit Learning.* Neural Monkey can be also used for bandit learning using expected loss minimization (Kreutzer et al., 2017). In this setup, a pre-trained model is tuned using either a simulated or real user feedback.
- *Detailed Logging and Visualization.* We implement several evaluation metrics that can be used for continuous validation of the models. We also plot the loss, norms of the parameters, and histograms of the parameter gradients to TensorBoard. It can also be used for embeddings visualization. There is also a standalone tool for attention visualization (Rikters et al., 2017).

4 Research Contributions

The development of Neural Monkey began in spring 2016 and since that time it has been used in various research papers and MT competitions. In this section, we describe a selection of these contributions.

Kreutzer et al. (2017) used Neural Monkey for domain adaptation using bandit learning with simulated user feedback. This technique brings a significant improvement over both the out-of-domain model and other domain adaptation techniques.

Libovický and Helcl (2017) published interpretable attention combination strategies for multi-source sequence-to-sequence learning tasks, such as multimodal translation. These combination strategies allow to explicitly model the different importance of the source sequences in each step during the decoding.

Bastings et al. (2017) used Neural Monkey to develop a new convolutional architecture for encoding the input sentences using dependency trees. This syntax-aware source language representation brings a consistent improvement over the attentive sequence-to-sequence model baseline.

Kocmi and Bojar (2017) published a thorough examination of various embedding initialization strategies, both random and pre-trained. They conclude that improvements from pre-trained embeddings are not always beneficial, and that high-quality embeddings can be trained using an initialization with a normal distribution with a small variance or even with zeros.

Neural Monkey has also been used in various submissions to shared tasks at the Conference on Machine Translation (WMT). In 2016 and 2017, it was used in submissions for multimodal translation task (Libovický et al., 2016; Helcl and Libovický, 2017). It was also part of a system combination used in the English-to-Czech MT system in the news task (Sudarikov et al., 2017). Neural Monkey models were used as baselines in bandit learning and neural training tasks. (Sokolov et al., 2017; Bojar et al., 2017b).

5 Comparison to Other Toolkits

Due to its goals, Neural Monkey differs from other open-source packages used for sequence-to-sequence learning.[3]

There are several production-oriented tools. They aim to be easy to use and work well in an online setup. They also aim to reach the best result possible as long as it does not intervene with its practical usability. Tools like this are most importantly *OpenNMT* (Klein et al., 2017), *Marian* (Junczys-Dowmunt et al., 2016), *Sockeye NMT* (Hieber et al., 2017), and *Tensor2Tensor* (Kaiser et al., 2017). All of them contain several well-tuned pre-made architectures that can be used out of the box for sequence-to-sequence learning tasks.

Tools like *Nematus* (Sennrich et al., 2017b) and *nmtpy* (Caglayan et al., 2017) built on Theano (Al-Rfou et al., 2016) or *xnmt* based on DyNet (Neubig et al., 2017) are more research oriented. For instance, Nematus implements several techniques which improve the performance of the system in the offline setup but which would be prohibitively slow in the online use. Using these techniques (such as model ensembling or right-to-left rescoring) repeatedly achieved the best results in the annual MT systems comparison at WMT (Sennrich et al., 2017a; Bojar et al., 2016, 2017a).

Research-oriented tools allow easy modification of the model architectures and other model properties. Unlike Neural Monkey, these software packages do not declare an ambition to collect architectures for other NLP tasks, nor do they plan to drive research towards understanding representations learned by the networks.

6 Conclusions and Future Work

The main future plans with Neural Monkey remain unchanged: the implementation of the state-of-the-art neural architectures for NMT with a strong focus on multimodality in MT.

Among the important features we plan to implement are the support for multi-GPU training via data parallelism and model import from other tools heavily used in MT research, most importantly Tensor2Tensor and Marian.

Additionally, we plan to focus on understanding language representations that emerge in neural architectures during end-to-end training of NLP tasks. In order to do so, we would like to get close to the state-of-the-art in other NLP tasks including image captioning, sentence classification tasks (e.g. sentiment analysis), or various sequence labeling tasks (e.g. tagging, named entity recognition). For studying the representations, we will implement additional visualization tools.

With the growing number of implemented models, we plan to introduce a model zoo. Neural Monkey users will be able to reuse configuration files, data preprocessing scripts, or even trained models or their parts in their own research.

Acknowledgements

This research received support from the Czech Science Foundation grant no. P103/12/G084, the EU grant no. H2020-ICT-2014-1-645452 (QT21), and the Charles University grant no. 8502/2016.

References

Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G. S., Davis, A., Dean, J., Devin, M., et al. (2016). Tensorflow: Large-scale machine learning on heterogeneous distributed systems. *arXiv preprint arXiv:1603.04467*.

[3] For an up-to-date list, see `https://github.com/jonsafari/nmt-list`.

Al-Rfou, R., Alain, G., Almahairi, A., Angermüller, C., Bahdanau, D., Ballas, N., Bastien, F., Bayer, J., Belikov, A., Belopolsky, A., Bengio, Y., Bergeron, A., Bergstra, J., Bisson, V., Snyder, J. B., Bouchard, N., Boulanger-Lewandowski, N., Bouthillier, X., de Brébisson, A., Breuleux, O., Carrier, P. L., Cho, K., Chorowski, J., Christiano, P. F., Cooijmans, T., Côté, M., Côté, M., Courville, A. C., Dauphin, Y. N., Delalleau, O., Demouth, J., Desjardins, G., Dieleman, S., Dinh, L., Ducoffe, M., Dumoulin, V., Kahou, S. E., Erhan, D., Fan, Z., Firat, O., Germain, M., Glorot, X., Goodfellow, I. J., Graham, M., Gülçehre, Ç., Hamel, P., Harlouchet, I., Heng, J., Hidasi, B., Honari, S., Jain, A., Jean, S., Jia, K., Korobov, M., Kulkarni, V., Lamb, A., Lamblin, P., Larsen, E., Laurent, C., Lee, S., Lefrançois, S., Lemieux, S., Léonard, N., Lin, Z., Livezey, J. A., Lorenz, C., Lowin, J., Ma, Q., Manzagol, P., Mastropietro, O., McGibbon, R., Memisevic, R., van Merriënboer, B., Michalski, V., Mirza, M., Orlandi, A., Pal, C. J., Pascanu, R., Pezeshki, M., Raffel, C., Renshaw, D., Rocklin, M., Romero, A., Roth, M., Sadowski, P., Salvatier, J., Savard, F., Schlüter, J., Schulman, J., Schwartz, G., Serban, I. V., Serdyuk, D., Shabanian, S., Simon, É., Spieckermann, S., Subramanyam, S. R., Sygnowski, J., Tanguay, J., van Tulder, G., Turian, J. P., Urban, S., Vincent, P., Visin, F., de Vries, H., Warde-Farley, D., Webb, D. J., Willson, M., Xu, K., Xue, L., Yao, L., Zhang, S., and Zhang, Y. (2016). Theano: A python framework for fast computation of mathematical expressions. *CoRR*, abs/1605.02688.

Bahdanau, D., Cho, K., and Bengio, Y. (2014). Neural machine translation by jointly learning to align and translate. *CoRR*, abs/1409.0473.

Bastings, J., Titov, I., Aziz, W., Marcheggiani, D., and Simaan, K. (2017). Graph convolutional encoders for syntax-aware neural machine translation. pages 1957–1967.

Bojar, O., Chatterjee, R., Federmann, C., Graham, Y., Haddow, B., Huang, S., Huck, M., Koehn, P., Liu, Q., Logacheva, V., Monz, C., Negri, M., Post, M., Rubino, R., Specia, L., and Turchi, M. (2017a). Findings of the 2017 conference on machine translation (wmt17). In *Proceedings of the Second Conference on Machine Translation*, pages 169–214, Copenhagen, Denmark. Association for Computational Linguistics.

Bojar, O., Chatterjee, R., Federmann, C., Graham, Y., Haddow, B., Huck, M., Yepes, A., Koehn, P., Logacheva, V., Monz, C., Negri, M., Névéol, A., Neves, M., Popel, M., Post, M., Rubino, R., Scarton, C., Specia, L., Turchi, M., Verspoor, K., and Zampieri, M. (2016). Findings of the 2016 conference on machine translation (WMT16). In *Proceedings of the First Conference on Machine Translation (WMT). Volume 2: Shared Task Papers*, volume 2, pages 131–198, Stroudsburg, PA, USA. Association for Computational Linguistics, Association for Computational Linguistics.

Bojar, O., Helcl, J., Kocmi, T., Libovický, J., and Musil, T. (2017b). Results of the wmt17 neural mt training task. In *Proceedings of the Second Conference on Machine Translation, Volume 2: Shared Task Papers*, pages 525–533, Copenhagen, Denmark. Association for Computational Linguistics.

Caglayan, O., García-Martínez, M., Bardet, A., Aransa, W., Bougares, F., and Barrault, L. (2017). Nmtpy: A flexible toolkit for advanced neural machine translation systems. *Prague Bull. Math. Linguistics*, 109:15–28.

Cho, K., van Merrienboer, B., Bahdanau, D., and Bengio, Y. (2014). On the properties of neural machine translation: Encoder–decoder approaches. In *Proceedings of SSST-8, Eighth Workshop on Syntax, Semantics and Structure in Statistical Translation*, pages 103–111, Doha, Qatar. Association for Computational Linguistics.

Do, T.-M.-T. and Artieres, T. (2010). Neural conditional random fields. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, volume 9. JMLR: W&CP.

Firat, O. and Cho, K. (2016). Conditional gated recurrent unit with attention mechanism. https://github.com/nyu-dl/dl4mt-tutorial/blob/master/docs/cgru.pdf. Published online, version `adbaeea`.

Gehring, J., Auli, M., Grangier, D., and Dauphin, Y. (2017). A convolutional encoder model for neural machine translation. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 123–135. Association for Computational Linguistics.

Graves, A., Fernández, S., Gomez, F., and Schmidhuber, J. (2006). Connectionist temporal classification: Labelling unsegmented sequence data with recurrent neural networks. In *Proceedings of the 23rd International Conference on Machine Learning*, ICML '06, pages 369–376, New York, NY, USA. ACM.

He, K., Zhang, X., Ren, S., and Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778.

Helcl, J. and Libovický, J. (2017). Cuni system for the wmt17 multimodal translation task. In *Proceedings of the Second Conference on Machine Translation*, pages 450–457, Copenhagen, Denmark. Association for Computational Linguistics.

Helcl, J. and Libovický, J. (2017). Neural monkey: An open-source tool for sequence learning. *The Prague Bulletin of Mathematical Linguistics*, (107):5–17.

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.

Hochreiter, S. and Schmidhuber, J. (1997). Long short-term memory. *Neural Comput.*, 9:1735–1780.

Junczys-Dowmunt, M., Dwojak, T., and Hoang, H. (2016). Is neural machine translation ready for deployment? a case study on 30 translation directions. In *Proceedings of the 9th International Workshop on Spoken Language Translation (IWSLT)*, Seattle, WA.

Kaiser, L., Gomez, A. N., Shazeer, N., Vaswani, A., Parmar, N., Jones, L., and Uszkoreit, J. (2017). One model to learn them all. *CoRR*, abs/1706.05137.

Kim, Y. (2014). Convolutional neural networks for sentence classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1746–1751, Doha, Qatar. Association for Computational Linguistics.

Klein, G., Kim, Y., Deng, Y., Senellart, J., and Rush, A. M. (2017). Opennmt: Open-source toolkit for neural machine translation. *CoRR*, abs/1701.02810.

Kocmi, T. and Bojar, O. (2017). An Exploration of Word Embedding Initialization in Deep-Learning Tasks. In *Proceedings of the 14th International Conference on Natural Language Processing*.

Kreutzer, J., Sokolov, A., and Riezler, S. (2017). Bandit structured prediction for neural sequence-to-sequence learning. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1503–1513, Vancouver, Canada. Association for Computational Linguistics.

Lafferty, J. D., McCallum, A., and Pereira, F. C. N. (2001). Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the Eighteenth International Conference on Machine Learning*, ICML '01, pages 282–289, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

Lee, J., Cho, K., and Hofmann, T. (2017). Fully character-level neural machine translation without explicit segmentation. *Transactions of the Association for Computational Linguistics*, 5:365–378.

Libovický, J. and Helcl, J. (2017). Attention strategies for multi-source sequence-to-sequence learning. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, Vancouver, Canada. Association for Computational Linguistics.

Libovický, J., Helcl, J., Tlustý, M., Bojar, O., and Pecina, P. (2016). CUNI system for WMT16 automatic post-editing and multimodal translation tasks. In *Proceedings of the First Conference on Machine Translation*, pages 646–654, Berlin, Germany. Association for Computational Linguistics.

Lin, Z., Feng, M., dos Santos, C. N., Yu, M., Xiang, B., Zhou, B., and Bengio, Y. (2017). A structured self-attentive sentence embedding. In *International Conference on Learning Representations 2017 (Conference Track)*.

Lu, J., Xiong, C., Parikh, D., and Socher, R. (2016). Knowing when to look: Adaptive attention via a visual sentinel for image captioning. *CoRR*, abs/1612.01887.

Neubig, G., Dyer, C., Goldberg, Y., Matthews, A., Ammar, W., Anastasopoulos, A., Ballesteros, M., Chiang, D., Clothiaux, D., Cohn, T., Duh, K., Faruqui, M., Gan, C., Garrette, D., Ji, Y., Kong, L., Kuncoro, A., Kumar, G., Malaviya, C., Michel, P., Oda, Y., Richardson, M., Saphra, N., Swayamdipta, S., and Yin, P. (2017). Dynet: The dynamic neural network toolkit. *CoRR*, abs/1701.03980.

Rikters, M., Fishel, M., and Bojar, O. (2017). Visualizing neural machine translation attention and confidence. *The Prague Bulletin of Mathematical Linguistics*, 109(1):39–50.

Sennrich, R., Birch, A., Currey, A., Germann, U., Haddow, B., Heafield, K., Miceli Barone, A. V., and Williams, P. (2017a). The university of edinburgh's neural mt systems for wmt17. In *Proceedings of the Second Conference on Machine Translation*, pages 389–399, Copenhagen, Denmark. Association for Computational Linguistics.

Sennrich, R., Firat, O., Cho, K., Birch, A., Haddow, B., Hitschler, J., Junczys-Dowmunt, M., Läubli, S., Miceli Barone, A. V., Mokrý, J., and Nadejde, M. (2017b). Nematus: a toolkit for neural machine translation. In *Proceedings of the Software Demonstrations of the 15th Conference of the European Chapter of the Association for Computational Linguistics*, pages 65–68, Valencia, Spain. Association for Computational Linguistics.

Simonyan, K. and Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. *CoRR*, abs/1409.1556.

Sokolov, A., Kreutzer, J., Sunderland, K., Danchenko, P., Szymaniak, W., Fürstenau, H., and Riezler, S. (2017). A shared task on bandit learning for machine translation. In *Proceedings of the 2nd Conference on Machine Translation (WMT)*, Copenhagen, Denmark.

Sudarikov, R., Mareček, D., Kocmi, T., Variš, D., and Bojar, O. (2017). CUNI submission in WMT17: Chimera goes neural. In *Proceedings of the Second Conference on Machine Translation*, pages 248–256, Copenhagen, Denmark. 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.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., 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 6000–6010. Curran Associates, Inc.