Small Batch Sizes Improve Training of Low-Resource Neural MT

Àlex R. Atrio[1,2] and **Andrei Popescu-Belis**[1,2]

[1]HEIG-VD / HES-SO
Yverdon-les-Bains
Switzerland

[2]EPFL
Lausanne
Switzerland

{alejandro.ramirezatrio, andrei.popescu-belis}@heig-vd.ch

Abstract

We study the role of an essential hyper-parameter that governs the training of Transformers for neural machine translation in a low-resource setting: the batch size. Using theoretical insights and experimental evidence, we argue against the widespread belief that batch size should be set as large as allowed by the memory of the GPUs. We show that in a low-resource setting, a smaller batch size leads to higher scores in a shorter training time, and argue that this is due to better regularization of the gradients during training.

1 Introduction

Training Transformers for low-resource neural machine translation (NMT), i.e. when only small parallel corpora are available, raises the challenge of finding optimal hyper-parameters. While several fixed configurations of the Transformer (Vaswani et al., 2017) have been empirically validated by the community, such as 'Base' or 'Big', the settings of many other hyper-parameters rely on tips from practitioners. However, these values are not always suitable to low-resource settings, and systematic studies in these settings are rare (Araabi and Monz, 2020; Van Biljon et al., 2020).

In this paper, we show that the best values of a hyper-parameter that is essential for training, namely *batch size*, differ in low-resource settings from those commonly accepted when larger data sets are available. We analyze the role of small batch sizes, inspired by studies in computer vision (Keskar et al., 2016), and then pinpoint empirically the optimal trade-off between a high batch size (for efficiency) and a small one (for regularization). Although large batch sizes were found to lead to higher-quality models in experiments with high-resource NMT (Popel and Bojar, 2018; Xu et al., 2020), we show here that smaller batch sizes can outperform the latter, likely due to a regularizing effect in the gradient update. Moreover, we show that this finding is invariant to changes in tokenization methods.

The paper is organized as follows. In Section 2, we discuss batch size from a machine learning perspective, showing why smaller values of batch size may act as regularizers. Then, in Section 3, we review studies of hyper-parameters in NMT. In Section 4, we present the parameters of our Transformer and the data from the WMT 2020 Low-resource task (Fraser, 2020) and other sources that we use in our experiments. In Section 5, we provide empirical evidence that smaller batch sizes are preferable in low-resource settings.

2 ML Perspective on Batch Size

Machine learning theory argues that performing back-propagation with large batch sizes leads to better optimization, because the estimates of the gradients are more accurate. Conversely, using small batches during training leads to noisier gradient estimations, i.e. with a larger variance in comparison to the gradient computed over the entire training set. Still, one advantage of small batch sizes is that they are more likely to make parameters converge towards flatter minima of the loss (Goodfellow et al., 2016, Chapter 8.1.3), as explained below. Such flatter minima have better generalization capacities, i.e. they maintain performance when presented with a new test set.

Keskar et al. (2016) define a *flat minimizer* – as opposed to a *sharp* one – as a point in the parameter space that is a local minimum of the loss function, and where this function varies slowly in a relatively large neighborhood. Keskar et al. (2016) point to the following *generalization gap*: training with large batch sizes tends to converge towards sharp minimizers, which offer poorer generalization ca-

Proceedings of the 18th International Conference on Natural Language Processing, pages 18–24
Silchar, India. December 16 - 19, 2021.

pacities. Conversely, *small batch sizes allow convergence towards flat minimizers*, which are likely to generalize better. Thus, smaller batch sizes have *exploration abilities*: the search is more likely to exit the basins of sharp minimizers, and to tend towards flat minimizers, from where noise will not cause them to exit.

Since a sharp minimizer requires high precision to be described, unlike a flat one, the more noise there is in the gradient, the more unlikely it is that the parameters will converge towards a sharp minimizer. This is precisely the contribution of a smaller batch size: introduce noise in the gradient estimation. According to this theoretical view, above a certain threshold of the batch size, the generalization capacities of a model deteriorate. The threshold depends on several hyper-parameters, including the batch size. Its role has not been fully settled yet, with observations and conclusions varying widely across studies (Dinh et al., 2017; Hoffer et al., 2017; Goyal et al., 2017; Li et al., 2017; Kawaguchi et al., 2017). Moreover, these studies are on image data sets, with fully connected or with convolutional NNs, which differ substantially from NMT settings.

3 The Role of Batch Size in Neural MT

Several recent studies in NMT have considered batch size among other hyper-parameters, but they have either been in high-resource settings (Popel and Bojar, 2018; Xu et al., 2020) or have given only marginal attention to batch size (Sennrich and Zhang, 2019; Araabi and Monz, 2020).

Popel and Bojar (2018) reported that BLEU scores increased with batch size (including when using more GPUs) in a Transformer-based NMT system, although with diminishing returns, recommending in particular that "batch size should be set as high as possible". Their experiments were performed using mainly two datasets, with respectively 58M and 15M sentence pairs. It thus remains an open question whether their findings regarding batch size also apply when much less training data is available.

Sennrich and Zhang (2019) experimented with a recurrent network in a low-resource setting and found that smaller batch sizes were beneficial, along with other forms of regularization. They experimented with two batch sizes of 4,000 and 1,000 tokens, and observed improvements with the latter of 0.30 and 0.04 BLEU points on data sets with 5k and 160k sentence pairs, respectively. It is difficult to predict from these results what the optimal batch size is for Transformer-based NMT.

Araabi and Monz (2020) studied the role of 15 hyper-parameters of the Transformer, with several sizes of low-resource datasets. For the largest training sizes tested (80k and 165k sentence pairs), larger batch sizes improved performance, with respectively 8,192 and 12,288 versus 4,096 for the other sizes. For lower training sizes, smaller batch sizes did not improve performance, which the authors explain by Transformer's need for larger batches. In our view, an alternative explanation is the order of optimization of the hyper-parameters (a grid search in which they optimize one hyper-parameter at a time): batch size is #12 out of 15, so by the time several sizes are compared, regularization has already been introduced in the model by dropouts on words, activation, and layers. Late optimization of batch size, of warmup steps (#14) or of learning rate (#15) cannot properly determine their regularizing effects.

Xu et al. (2020) proposed to compute gradients while accumulating minibatches, and observed that increasing batch size stabilizes gradient direction up to a certain point, after which it starts to fluctuate. They used this criterion to dynamically adjust batch sizes while training. In their experiments with large training sets (4.5M and 36M sentence pairs), their average batch size was around 26k on two GPUs, and never lower than 7k. Their observations on the gradient direction as more minibatches are accumulated are consistent with the findings of Popel and Bojar (2018) who see diminishing returns when increasing batch size.

4 Datasets and Systems

We train NMT systems with two low-resource parallel corpora, listed in the first two lines of Table 1: the Upper Sorbian (HSB) to German (DE) training data of the WMT 2020 Low-Resource Translation Task (Fraser, 2020) and a low-size excerpt of the German to English News Commentary v13 (Bojar et al., 2018), from which we randomly sampled 60k parallel lines. For the HSB-DE models, we also use the development and test sets provided by the WMT 2020 and 2021 Low-Resource Translation Task (Libovický and Fraser, 2021), each consisting of 2k sentences, and for DE-EN we sample a development set and a test set from the original corpus, with 2k sentences each as well. We apply a com-

Dataset	Lang.	Orig.	Filt.	Δ%
WMT20 Low-res.	HSB-DE	60k	59.8k	0.29
News Comm. v13	DE-EN	60k	59.9k	0.20
Sorbian Institute	HSB	339k	339k	0.00
Witaj	HSB	222k	220k	0.84
Web	HSB	134k	121k	9.98
Europarl v8	DE	2.2M	2.2M	0.79
News Comm. v15	DE	422k	411k	2.58
JW300	DE	2.3M	2.2M	4.44
Europarl v3	DE	790k	785k	0.69
Europarl v3	EN	790k	782k	1.07

Table 1: Numbers of lines in the original and filtered corpora used in our experiments. HSB stands for Upper Sorbian and Δ% for the proportion of lines filtered out. The only *parallel* corpora used for training NMT are the first two ones; the other corpora are only used to train the SentencePiece model.

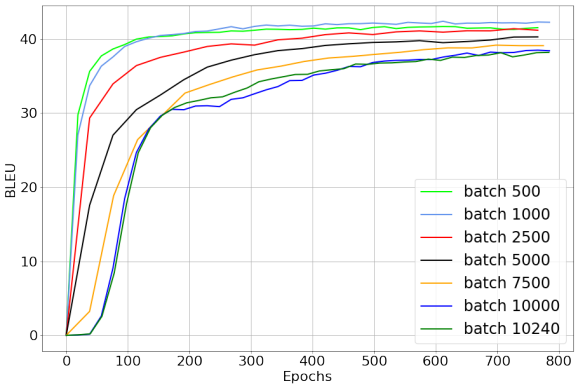

Figure 1: BLEU scores on the test set for HSB-DE models trained with different batch sizes.

mon filtering process for all data used: we delete from all our data the sentences that are not between 2 and 300 words long, with resulting numbers of lines shown in Table 1.

We build subword vocabularies using the Unigram LM model (Sennrich et al., 2016; Kudo, 2018) as implemented in SentencePiece[1], with the monolingual corpora from Table 1. We train a shared model for HSB-DE with a vocabulary of 32k pieces, character coverage of 0.98, *nbest* = 1 and *alpha* = 0. The HSB data adds up to 740k sentences, and we sample 680k sentences from three DE corpora, and add them to the 60k sentences from the DE side of the parallel HSB-DE corpus. To train the SentencePiece model for the DE-EN, for comparison purposes, we treat German as a low-resource language, and sample 680k lines of English and German from Europarl v3 (Tiedemann, 2012), which we combine respectively with the 60k lines extracted from the DE-EN parallel corpus.

We use the Transformer-Base (Vaswani et al., 2017) in the implementation provided by OpenNMT (Klein et al., 2017, 2020), with the parameters given in Appendix A. Unless otherwise specified, we follow OpenNMT-py's recommended values for the hyper-parameters.[2]

When using several GPUs with gradient accumulation, each GPU processes several batches, which are then accumulated across all GPUs and used to update the model at each step. Therefore, the *effective batch size* is $B \times G \times A$, where B is the individual batch size, G is the number of GPUs and A the number of accumulated batches, and differs from the `batch_size` hyper-parameter B. We train all models on two GeForce RTX 1080Ti GPUs with 11 GB of memory each and accumulate gradients over two minibatches ($A = 2$), following OpenNMT-py's recommendation. Therefore, the `batch_size` parameter is not our effective batch size, which is four times larger. Throughout this work, we will refer to batch size B as the `batch_size` parameter, and report true *epochs*, which we define as computed with the effective batch size as $S \times B_{eff}/N$, for S training steps, B_{eff} effective batch size, and N number of source tokens in the training set.

Following OpenNMT-py's recommendations, we set the Adam hyper-parameters at $\beta_1 = 0.9$, $\beta_2 = 0.998$, $\epsilon = 10^{-8}$ and apply at each step a scaling factor of two to Noam's learning rate schedule, setting warmup steps to 8k. Translations are generated with a beam width of seven, with an ensemble of the last four saved checkpoints. We report BLEU scores (Papineni et al., 2002) obtained with SacreBLEU (Post, 2018) on detokenized text.

5 Experimental Results

To study the impact of batch sizes in a low-resource setting, we train various HSB-DE and DE-EN models for 700 epochs with the following batch sizes: 100, 250, 500, 1,000, 2,500, 5,000, 7,500, 10,000, and 10,240 (this is the largest one that fits in our GPU memory).

5.1 NMT Performance

NMT performance on the HSB-DE test set throughout the training is shown in Figure 1, with BLEU scores depending on the number of epochs. The evolution depending on real training time (wall

[1] https://github.com/google/sentencepiece

[2] https://opennmt.net/OpenNMT-py/examples/Translation.html

Batch Size	HSB-DE								DE-EN							
	dev				test				dev				test			
	Xent	BLEU	chrF	TER	Xent	BLEU	chrF	TER	Xent	BLEU	chrF	TER	Xent	BLEU	chrF	TER
500	0.03	48.12	71.13	37.35	0.03	41.53	67.34	43.84	0.11	37.35	58.04	54.54	0.11	37.72	58.35	54.60
1,000	**0.02**	**49.23**	**72.07**	**36.35**	**0.02**	**42.26**	**67.93**	**43.16**	0.05	**38.03**	**59.39**	**52.91**	0.05	**38.67**	**59.68**	**52.71**
2,500	0.03	48.28	71.63	37.02	0.03	41.18	67.36	44.02	**0.04**	33.83	56.70	56.27	**0.04**	35.51	57.76	55.47
5,000	0.03	46.99	70.74	38.05	0.03	40.28	66.62	45.24	0.05	32.47	55.20	57.88	0.05	33.97	56.16	57.08
7,500	0.03	46.05	70.29	38.87	0.03	39.10	65.94	46.18	0.05	32.67	55.99	57.67	0.05	33.80	56.72	57.21
10,000	0.04	44.61	69.19	40.00	0.04	38.41	65.67	46.45	0.05	31.84	55.20	58.35	0.05	33.50	56.14	57.63
10,240	0.04	45.59	70.12	39.26	0.04	38.19	65.39	46.79	0.06	31.49	55.00	58.65	0.06	33.03	55.78	58.07

Table 2: Loss and scores for models trained for 700 epochs with various batch sizes for HSB-DE and DE-EN directions. All differences in BLEU on the dev and test sets are statistically significant at the 95% level, except for the pairs in similar colors.

time) is similar in terms of rankings. Thus, the following analysis holds whether we train the models for the same amount of epochs or of hours.

The final scores on the development and test sets are given in Table 2, sorted by batch sizes. We provide first the actual loss of the model ('Xent' for cross-entropy), and then three typical NMT scores: BLEU (Papineni et al., 2002), chrF (Popović, 2015) and Translation Error Rate (Snover et al., 2006). The 100 and 250 batch size models did not reach BLEU scores significantly above zero, and are not included among the results in the table.

We test the statistical significance of the differences between each score and the others, with 95% confidence, using the paired bootstrap resampling tool from SacreBLEU (Post, 2018).[3] All differences between higher and lower BLEU scores are statistically significant, except the pairs highlighted in similar colors in Table 2.[4] The best NMT scores, which are always obtained with a batch size of 1,000, are significantly higher than all the other ones, including those obtained with the largest possible batch sizes for our GPU (10,000 or 10,240). We thus select two values for further experiments: a batch size of 1,000 as our highest-scoring model, and one of 10,000 as the maximum allowed by our GPU memory. A simple ratio of 10 holds between the two values.

These empirical results are contrary to those from Popel and Bojar (2018), who observe that increasing the batch size for Transformer-Base produces higher scores, although with diminishing returns after a certain threshold. We hypothesize that the main explanation is the difference between the amounts of training data: in our low-resource setting, we use 60k sentences, while Popel and Bojar (2018) use 57M sentences. Our findings are consistent with those of Keskar et al. (2016), who also observe that the optimal batch size is at the lower end of the range, on a computer vision task with convolutional and fully-connected NNs.

5.2 Asymptotic Performance

An alternative explanation for the previous results is that the learning rate is too small for the larger batch sizes, which require more time to converge. To test whether the differences observed above between small and large batch sizes depend on the actual training time, we continue training the 1,000 and 10,000 batch size models for HSB-DE and DE-EN for twice as many epochs as above (1400). The BLEU scores and their increases with respect to training for 700 epochs are given in Table 3. The performance gap (from +3.85 to +3.25 BLEU) between small and large batch sizes is not overturned by training the models for much longer.

The scores from our best system (1,000 batch size, 42.81 BLEU on the test set) are similar to scores obtained by *baselines* of the five highest-scoring teams at the WMT20 Low-resource shared task on HSB-DE (Fraser, 2020). While the scores of Scherrer et al. (2020) and Li et al. (2020) are not comparable due to a different architecture or the use of unsupervised pre-training, the baseline scores of Knowles et al. (2020), Libovický et al. (2020) and Kvapilíková et al. (2020) are respectively 44.1, 43.4, and 38.7. The first one is higher than our best BLEU by 1.29, likely due to the use of 43M lines of CS and DE data for the subword vocabulary, vs. 700k in our case.

[3] github.com/mjpost/sacrebleu with the signature nrefs:1|bs:1000|seed:12345|case:mixed|eff:no|tok:13a|smooth:exp|version:2.0.0.

[4] The difference in BLEU between the following pairs is not significant. For HSB-DE, 2,500 vs. 500, and 10,240 vs. 10,000, on the test set; and 2,500 vs. 500, and 7,500 vs. 10,240 on the dev set. For DE-EN these are 7,500 vs. 5,000, and 10,000 vs. 7,500, on the test set; and 1,000 vs. 500, 5,000 vs. 7,500, and 10,000 vs 10,240, on the dev set.

Batch size	HSB-DE		DE-EN	
	dev	test	dev	test
1,000	49.52 (*+0.29*)	42.81 (*+0.55*)	38.67 (*+0.64*)	39.24 (*+0.57*)
10,000	46.44 (*+1.83*)	39.56 (*+1.15*)	33.19 (*+1.35*)	34.42 (*+0.92*)

Table 3: BLEU scores for models trained for 1,400 epochs. The scores for 1,000 are significantly higher (at 95%) than those for 10,000. In parenthesis, the absolute difference with BLEU scores after 700 epochs.

5.3 Invariance with respect to Vocabulary

We additionally perform two comparisons that show that the above results hold regardless of the tokenizer and the vocabulary size. First, we test whether the score difference is preserved with an unshared SentencePiece vocabulary, i.e. when not sharing the source (HSB and DE) and the target (DE and EN) vocabularies.

Second, we train two NMT models for HSB-DE using a Byte Pair Encoding (BPE) vocabulary (Sennrich et al., 2016), which we generate using the `learn_bpe.py` tool from OpenNMT-py, with 32k merge operations and the remaining parameters at default values. Table 4 shows BLEU scores on the development sets for batch sizes of 1,000 and 10,000. The previously observed differences in score between the batch sizes still hold, and we see that a shared SentencePiece vocabulary leads to a better NMT system than an unshared or a BPE one.

Batch size	SP unshared		BPE
	HSB-DE	DE-EN	HSB-DE
1,000	46.80 (*-2.43*)	35.90 (*-2.13*)	46.21 (*-3.02*)
10,000	41.99 (*-2.62*)	30.09 (*-1.75*)	43.35 (*-1.26*)

Table 4: BLEU scores on the dev set for HSB-DE and DE-EN models trained with SentencePiece (SP) vocabularies not shared between source and target (left) and BPE subwords (right). The scores for 1,000 are significantly higher (at 95%) than those for 10,000. In parenthesis, the difference with BLEU scores obtained with the SP shared vocabulary.

6 Conclusion and Future Work

In this work, we have shown that insights from computer vision on the regularizing effect of small batch sizes are also applicable to NMT. Our results, focused on a low-resource setting, challenge those of previous NMT studies with large amounts of training data, and the general belief that batch sizes should be as large as they fit in the GPU memory. We have shown that training with small batch sizes leads to models that generalize better, and found the optimal batch size below which performance degrades.

Future work should explore how the learning rate must be adjusted depending on the batch size, and whether a dynamically scheduled combination of batch size and learning rate can provide an even better regularizer. For instance, it should be tested if dynamic batch sizes as proposed by Xu et al. (2020) can also improve performance in a low-resource setting, with batch size thresholds changed to measure an optimal level of noise.

Acknowledgments

We thank for their financial support the Swiss National Science Foundation (grant n. 175693 for the DOMAT project: On-demand Knowledge for Document-level Machine Translation) and Armasuisse (FamilyMT project). We especially thank Dr. Ljiljana Dolamic (Armasuisse) for her support in the FamilyMT project. We are also grateful to the anonymous reviewers and to Giorgos Vernikos for their helpful suggestions.

References

Ali Araabi and Christof Monz. 2020. Optimizing Transformer for low-resource neural machine translation. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 3429–3435, Barcelona, Spain (Online). International Committee on Computational Linguistics.

Ondřej Bojar, Christian Federmann, Mark Fishel, Yvette Graham, Barry Haddow, Philipp Koehn, and Christof Monz. 2018. Findings of the 2018 conference on machine translation (WMT18). In *Proceedings of the Third Conference on Machine Translation: Shared Task Papers*, pages 272–303, Belgium, Brussels. Association for Computational Linguistics.

Laurent Dinh, Razvan Pascanu, Samy Bengio, and Yoshua Bengio. 2017. Sharp minima can generalize for deep nets. In *International Conference on Machine Learning*, pages 1019–1028. PMLR.

Alexander Fraser. 2020. Findings of the WMT 2020 shared tasks in unsupervised MT and very low resource supervised MT. In *Proceedings of the Fifth Conference on Machine Translation*, pages 765–771, Online. Association for Computational Linguistics.

Ian Goodfellow, Yoshua Bengio, and Aaron Courville. 2016. *Deep Learning*. MIT Press.

Priya Goyal, Piotr Dollár, Ross Girshick, Pieter Noordhuis, Lukasz Wesolowski, Aapo Kyrola, Andrew Tulloch, Yangqing Jia, and Kaiming He. 2017. Accurate, large minibatch SGD: Training Imagenet in 1 hour. *arXiv preprint arXiv:1706.02677*.

Elad Hoffer, Itay Hubara, and Daniel Soudry. 2017. Train longer, generalize better: Closing the generalization gap in large batch training of neural networks. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, page 1729–1739, Red Hook, NY, USA. Curran Associates Inc.

Kenji Kawaguchi, Leslie Pack Kaelbling, and Yoshua Bengio. 2017. Generalization in deep learning. *arXiv preprint arXiv:1710.05468*.

Nitish Shirish Keskar, Dheevatsa Mudigere, Jorge Nocedal, Mikhail Smelyanskiy, and Ping Tak Peter Tang. 2016. On large-batch training for deep learning: Generalization gap and sharp minima. *arXiv preprint arXiv:1609.04836*.

Guillaume Klein, François Hernandez, Vincent Nguyen, and Jean Senellart. 2020. The OpenNMT neural machine translation toolkit: 2020 edition. In *Proceedings of the 14th Conference of the Association for Machine Translation in the Americas (Volume 1: Research Track)*, pages 102–109, Virtual. Association for Machine Translation in the Americas.

Guillaume Klein, Yoon Kim, Yuntian Deng, Jean Senellart, and Alexander Rush. 2017. OpenNMT: Open-source toolkit for NMT. In *Proceedings of ACL 2017, System Demonstrations*, pages 67–72. Association for Computational Linguistics.

Rebecca Knowles, Samuel Larkin, Darlene Stewart, and Patrick Littell. 2020. NRC systems for low resource German-Upper Sorbian machine translation 2020: Transfer learning with lexical modifications. In *Proceedings of the Fifth Conference on Machine Translation*, pages 1112–1122, Online. Association for Computational Linguistics.

Taku Kudo. 2018. Subword regularization: Improving neural network translation models with multiple subword candidates. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 66–75.

Ivana Kvapilíková, Tom Kocmi, and Ondřej Bojar. 2020. CUNI systems for the unsupervised and very low resource translation task in WMT20. In *Proceedings of the Fifth Conference on Machine Translation*, pages 1123–1128, Online. Association for Computational Linguistics.

Hao Li, Zheng Xu, Gavin Taylor, Christoph Studer, and Tom Goldstein. 2017. Visualizing the loss landscape of neural nets. *arXiv preprint arXiv:1712.09913*.

Zuchao Li, Hai Zhao, Rui Wang, Kehai Chen, Masao Utiyama, and Eiichiro Sumita. 2020. SJTU-NICT's supervised and unsupervised neural machine translation systems for the WMT20 news translation task. In *Proceedings of the Fifth Conference on Machine Translation*, pages 218–229, Online. Association for Computational Linguistics.

Jindřich Libovický, Viktor Hangya, Helmut Schmid, and Alexander Fraser. 2020. The LMU Munich system for the WMT20 very low resource supervised MT task. In *Proceedings of the Fifth Conference on Machine Translation*, pages 1104–1111, Online. Association for Computational Linguistics.

Jindřich Libovický and Alexander Fraser. 2021. Findings of the WMT 2021 shared tasks in unsupervised MT and very low resource supervised MT. In *Proceedings of the Sixth Conference on Machine Translation*, Online. Association for Computational Linguistics.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318, Philadelphia, Pennsylvania, USA. Association for Computational Linguistics.

Martin Popel and Ondřej Bojar. 2018. Training tips for the Transformer model. *The Prague Bulletin of Mathematical Linguistics*, 110(1):43–70.

Maja Popović. 2015. chrF: character n-gram f-score for automatic MT evaluation. In *Proceedings of the Tenth Workshop on Statistical Machine Translation*, pages 392–395, Lisbon, Portugal. Association for Computational Linguistics.

Matt Post. 2018. A call for clarity in reporting BLEU scores. In *Proceedings of the Third Conference on Machine Translation: Research Papers*, pages 186–191, Belgium, Brussels. Association for Computational Linguistics.

Yves Scherrer, Stig-Arne Grönroos, and Sami Virpioja. 2020. The University of Helsinki and Aalto university submissions to the WMT 2020 news and low-resource translation tasks. In *Proceedings of the Fifth Conference on Machine Translation*, pages 1129–1138, Online. Association for Computational Linguistics.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. NMT 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.

Rico Sennrich and Biao Zhang. 2019. Revisiting low-resource neural machine translation: A case study. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 211–221, Florence, Italy. Association for Computational Linguistics.

Matthew Snover, Bonnie Dorr, Richard Schwartz, Linnea Micciulla, and John Makhoul. 2006. A study of translation edit rate with targeted human annotations. In *Proceedings of the Conference of the Association for Machine Translation in the Americas (AMTA 2006)*.

Jörg Tiedemann. 2012. Parallel data, tools and interfaces in OPUS. In *Proceedings of the Eight International Conference on Language Resources and Evaluation (LREC'12)*, Istanbul, Turkey. European Language Resources Association (ELRA).

Elan Van Biljon, Arnu Pretorius, and Julia Kreutzer. 2020. On optimal Transformer depth for low-resource language translation. *arXiv preprint arXiv:2004.04418*.

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 *Advances in Neural Information Processing Systems 30*, pages 5998–6008.

Hongfei Xu, Josef van Genabith, Deyi Xiong, and Qiuhui Liu. 2020. Dynamically adjusting Transformer batch size by monitoring gradient direction change. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 3519–3524, Online. Association for Computational Linguistics.

A Appendix

The hyper-parameters used to train our models are the following ones:

```
src_words_min_frequency: 2
tgt_words_min_frequency: 2
valid_batch_size: 200
max_generator_batches: 2
optim: adam
learning_rate: 2.0
adam_beta2: 0.998
decay_method: noam
accum_count: 2
warmup_steps: 8000
label_smoothing: 0.1
max_grad_norm: 0
param_init: 0
param_init_glorot: true
normalization: tokens
encoder_type: transformer
decoder_type: transformer
position_encoding: true
layers: 6
heads: 8
rnn_size: 512
word_vec_size: 512
transformer_ff: 2048
dropout: 0.1
batch_size: 1000
batch_type: tokens
```