Pre-trained Language Model Representations for Language Generation

Sergey Edunov*, **Alexei Baevski***, **Michael Auli**
Facebook AI Research
Menlo Park, CA
{edunov,abaevski,michaelauli}@fb.com

Abstract

Pre-trained language model representations have been successful in a wide range of language understanding tasks. In this paper, we examine different strategies to integrate pre-trained representations into sequence to sequence models and apply it to neural machine translation and abstractive summarization. We find that pre-trained representations are most effective when added to the encoder network which slows inference by only 14%. Our experiments in machine translation show gains of up to 5.3 BLEU in a simulated resource-poor setup. While returns diminish with more labeled data, we still observe improvements when millions of sentence-pairs are available. Finally, on abstractive summarization we achieve a new state of the art on the full text version of CNN-DailyMail. [1]

1 Introduction

Pre-training of language models has been shown to provide large improvements for a range of language understanding tasks (Peters et al., 2018; Radford et al., 2018; Phang et al., 2018; Devlin et al., 2018). The key idea is to train a large generative model on vast corpora and use the resulting representations on tasks for which only limited amounts of labeled data is available. Pre-training of sequence to sequence models has been previously investigated for text classification (Dai and Le, 2015) but not for text generation. In neural machine translation, there has been work on transferring representations from high-resource language pairs to low-resource settings (Zoph et al., 2016).

In this paper, we apply pre-trained representations from language models to language generation tasks that can be modeled by sequence to sequence architectures. Previous work on integrating language models with sequence to sequence models focused on the decoder network and added language model representations right before the output of the decoder (Gulcehre et al., 2015). We extend their study by investigating several other strategies such as inputting ELMo-style representations (Peters et al., 2018) or fine-tuning the language model (§2).

Our experiments rely on strong transformer-based language models trained on up to six billion tokens (§3). We present a detailed study of various strategies in different simulated labeled training data scenarios and observe the largest improvements in low-resource settings but gains of over 1 BLEU are still possible when five million sentence-pairs are available. The most successful strategy to integrate pre-trained representations is as input to the encoder network (§4).

2 Strategies to add representations

We consider augmenting a standard sequence to sequence model with pre-trained representations following an ELMo-style regime (§2.1) as well as by fine-tuning the language model (§2.2).

2.1 ELMo augmentation

The ELMo approach of Peters et al. (2018) forms contextualized word embeddings based on language model representations without adjusting the actual language model parameters. Specifically, the ELMo module contains a set of parameters $\lambda_1 \dots \lambda_L, \gamma$ to form a linear combination of the L layers of the language model: ELMo = $\gamma \sum_{i=0}^{L} \frac{1}{Z} \exp(\lambda_i) \mathbf{h}^k$ where γ is a learned scalar, Z is a constant to normalize the $\exp(\lambda_i)$ to sum to one and $\mathbf{h}^k$ is the output of the k-th language model layer; the module also considers the input word embeddings of the language model. We also

*Equal contribution.

[1] Code and pre-trained models are available at https://github.com/pytorch/fairseq/tree/bi_trans_lm/examples/pretraining

Proceedings of NAACL-HLT 2019, pages 4052–4059
Minneapolis, Minnesota, June 2 - June 7, 2019. ©2019 Association for Computational Linguistics

apply layer normalization (Ba et al., 2016) to each $\mathbf{h}^k$ before computing ELMo vectors.

We experiment with an ELMo module to input contextualized embeddings either to the encoder (SRC-ELMO) or the decoder (TGT-ELMO). This provides word representations specific to the current input sentence and these representations have been trained on much more data than is available for the text generation task.

2.2 Fine-tuning approach

Fine-tuning the pre-trained representations adjusts the language model parameters by the learning signal of the end-task (Radford et al., 2018; Devlin et al., 2018). We replace learned input word embeddings in the encoder network with the output of the language model (SRC-FT). Specifically, we use the language model representation of the layer before the softmax and feed it to the encoder. We also add dropout to the language model output. Tuning separate learning rates for the language model and the sequence to sequence model may lead to better performance but we leave this to future work. However, we do tune the number of encoder blocks N as we found this important to obtain good accuracy for this setting. We apply the same strategy to the decoder: we input language model representations to the decoder network and fine-tune the language model when training the sequence to sequence model (TGT-FT).

3 Experimental setup

3.1 Datasets

Pre-training. We train language models on two languages: One model is estimated on the German newscrawl distributed by WMT'18 comprising 260M sentences or 6B tokens. Another model is trained on the English newscrawl data comprising 193M sentences or 5B tokens. We learn a joint Byte-Pair-Encoding (BPE; Sennrich et al., 2016) vocabulary of 37K types on the German and English newscrawl and train the language models with this vocabulary.

Machine translation. We consider two benchmarks: Most experiments are run on the WMT'18 English-German (en-de) news translation task and we validate our findings on the WMT'18 English-Turkish (en-tr) news task. For WMT'18 English-German, the training corpus consists of all available bitext excluding the ParaCrawl corpus and we remove sentences longer than 250 tokens as well as sentence-pairs with a source/target length ratio exceeding 1.5. This results in 5.18M sentence pairs. We tokenize all data with the Moses tokenizer (Koehn et al., 2007) and apply the BPE vocabulary learned on the monolingual corpora.

For WMT'18 English-Turkish, we use all of the available bitext comprising 208K sentence-pairs without any filtering. We develop on newstest2017 and test on newstest2018. For en-tr we only experiment with adding representations to the encoder and therefore apply the language model vocabulary to the source side. For the target vocabulary we learn a BPE code with 32K merge operations on the Turkish side of the bitext. Both datasets are evaluated in terms of case-sensitive de-tokenized BLEU (Papineni et al., 2002; Post, 2018).[2]

Summarization. We consider the CNN-DailyMail abstractive document summarization task comprising over 280K news articles paired with multi-sentence summaries. CNN-DailyMail is a widely used dataset for abstractive text summarization. Following (See et al., 2017), we report results on the non-anonymized version of CNN-DailyMail rather than the entity-anonymized version (Hermann et al., 2015; Nallapati et al., 2016) because the language model was trained on full text. Articles are truncated to 400 tokens (See et al., 2017) and we use a BPE vocabulary of 32K types (Fan et al., 2017). We evaluate in terms of F1-Rouge, that is Rouge-1, Rouge-2 and Rouge-L (Lin, 2004).[3]

3.2 Language model pre-training

We consider two types of architectures: a bi-directional language model to augment the sequence to sequence encoder and a uni-directional model to augment the decoder. Both use self-attention (Vaswani et al., 2017) and the uni-directional model contains $N = 12$ transformer blocks, followed by a word classifier to predict the next word on the right. The bi-directional model solves a cloze-style token prediction task at training time (Baevski et al., 2019). The model consists of two towers, the *forward* tower operates left-to-right and the tower operating right-to-left as *backward* tower; each tower contains $N = 12$ trans-

[2] sacreBLEU signatures: BLEU+case.mixed+lang.en-{de,tr}+numrefs.1+smooth.exp+test.wmt18+tok.13a +version.1.2.1

[3] We use the following parameters for `ROUGE-1.5.5.pl`: -m -a -n 2

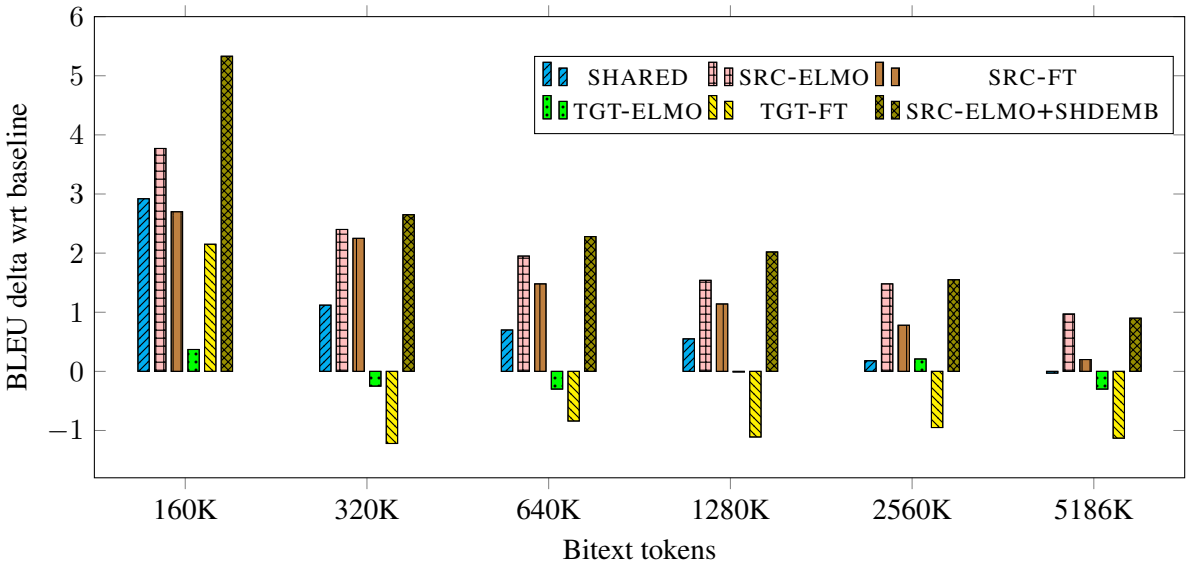

Figure 1: BLEU difference to a bitext-only baseline when adding pre-trained language model representations to a neural machine translation model in different simulated bitext settings. Results are based on averaging newstest2012-2017 of WMT English-German translation.

former blocks. The forward and backward representations are combined via a self-attention module and the output of this module is used to predict the token at position i. The model has access to the entire input surrounding the current target token. Models use the standard settings for the Big Transformer (Vaswani et al., 2017). The bi-directional model contains 353M parameters and the uni-directional model 190M parameters. Both models were trained for 1M steps using Nesterov's accelerated gradient (Sutskever et al., 2013) with momentum 0.99 following Baevski and Auli (2018). The learning rate is linearly warmed up from 10^{-7} to 1 for 16K steps and then annealed using a cosine learning rate schedule with a single phase to 0.0001 (Loshchilov and Hutter, 2016). We train on 32 Nvidia V100 SXM2 GPUs and use the NCCL2 library as well as the torch distributed package for inter-GPU communication. Training relies on 16-bit floating point operations (Ott et al., 2018) and it took six days for the bi-directional model and four days for the uni-directional model.

3.3 Sequence to sequence model

We use the transformer implementation of the fairseq toolkit (Ott et al., 2019). The WMT en-de and en-tr experiments are based on the Big Transformer sequence to sequence architecture with 6 blocks in the encoder and decoder. For abstractive summarization we use a base transformer model (Vaswani et al., 2017). We tune dropout values of between 0.1 and 0.4 on the validation set. Models are optimized with Adam (Kingma and Ba, 2015) using $\beta_1 = 0.9$, $\beta_2 = 0.98$, and $\epsilon = 1e-8$ and we use the same learning rate schedule as Vaswani et al. (2017); we perform 10K-200K depending on bitext size. All models use label smoothing with a uniform prior distribution over the vocabulary $\epsilon = 0.1$ (Szegedy et al., 2015; Pereyra et al., 2017). We run experiments on 8 GPUs and generate translations with a beam of size 5.

4 Results

4.1 Machine translation

We first present a comparison of the various strategies in different simulated parallel corpus size settings. For each experiment, we tune the dropout applied to the language model representations, and we reduce the number of optimizer steps for smaller bitext setups as models converge faster; all other hyper-parameters are equal between setups. Our baseline is a Big Transformer model and we also consider a variant where we share token embeddings between the encoder and decoder (SHARED; Inan et al., 2016; Press & Wolf, 2016).

Figure 1 shows results averaged over six test sets relative to the baseline which does not share source and target embeddings (Appendix A shows a detailed breakdown). SHARED performs very well with little labeled data but the gains erode to practically zero in large bitext settings. Pre-trained

	160K	640K	5186K
baseline	21.4	33.1	40.1
SRC-ELMO	26.6	35.6	41.8
SRC-FT	24.3	34.9	40.8
TGT-ELMO	21.3	31.9	40.5
TGT-FT	24.2	31.4	38.8
SRC-ELMO+SHDEMB	29.0	36.2	41.8

Table 1: BLEU on newstest2018 of WMT English-German in three simulated bitext size scenarios.

	news2017	news2018
baseline	9.8	9.5
SRC-ELMO	12.0	11.3
SRC-ELMO+SHDEMB	12.9	11.8

Table 2: WMT English-Turkish translation results in terms of BLEU on newstest2017 (valid) and newstest2018 (test) with ELMo inputs to the encoder.

language model representations are most effective in low bitext setups. The best performing strategy is ELMo embeddings input to the encoder (SRC-ELMO). This improves the baseline by 3.8 BLEU in the 160K bitext setting and it still improves the 5.2M setting by over 1 BLEU.

We further improve SRC-ELMO by sharing learned word representations in the decoder by tying input and output embeddings (SRC-ELMO+SHDEMB). This configuration performs even better than SRC-ELMO with a gain of 5.3 BLEU in the 160K setup. Sharing decoder embeddings is equally applicable to SRC-FT. Language model representations are much less effective in the decoder: TGT-FT improves the 160K bitext setup but yields no improvements thereafter and TGT-ELMO performs even worse. We conjecture that pre-trained representations give much easier wins in the encoder. Table 1 shows additional results on newstest2018.

Pre-trained representations mostly impacts the training time of the sequence to sequence model (see Appendix B): SRC-ELMO slows throughput during training by about 5.3x and SRC-FT is even slower because of the need to backpropagate through the LM for fine-tuning (9.2x). However, inference is only 12-14% slower than the baseline when adding pre-trained embeddings to the encoder (SRC-ELMO, SRC-FT). This is because the LM computation can be paralelized for all input tokens. Inference is much slower when adding representations to the decoder because the LM needs to be invoked repeatedly. Our current implementation does not cache LM operations for the previous state and can be made much faster.

	ROUGE		
	1	2	L
Lead-3	40.34	17.70	36.57
See et al. (2017)	39.53	17.28	36.38
Gehrmann et al. (2018)	41.22	18.68	38.34
baseline	40.07	17.61	36.78
SRC-ELMO+SHDEMB	41.56	18.94	38.47

Table 3: Abstractive summarization results on CNN-DailyMail. ELMo inputs achieve a new state of the art.

The baseline uses a BPE vocabulary estimated on the language model corpora (§3). Appendix A shows that this vocabulary actually leads to sligtly better performance than a joint BPE code learned on the bitext as is usual.

Next, we validate our findings on the WMT'18 English-Turkish task for which the bitext is truly limited (208K sentence-pairs). We use the language model vocab for the the English side of the bitext and a BPE vocabulary learned on the Turkish side. Table 2 shows that ELMo embeddings for the encoder improve English-Turkish translation.

4.2 Abstractive summarization

Following See et al. (2017), we experiment on the non-anonymized version of CNN-DailyMail. When generating summaries, we follow standard practice of tuning the maximum output length and disallow repeating the same trigram (Paulus et al., 2017; Fan et al., 2017). For this task we train language model representations on the combination of newscrawl and the CNN-DailyMail training data. Table 3 shows that pre-trained embeddings can significantly improve on top of a strong baseline transformer. We also compare to Gehrmann et al. (2018) who use a task-specific architecture compared to our generic sequence to sequence baseline. Pre-trained representations are complementary to their method.

5 Conclusion

We presented an analysis of different strategies to add pre-trained language model representations to sequence to sequence models for neural machine

translation and abstractive document summarization. Adding pre-trained representations is very effective for the encoder network and while returns diminish when more labeled data becomes available, we still observe improvements when millions of examples are available. In future research we will investigate ways to improve the decoder with pre-trained representations.

References

Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. 2016. Layer normalization. *arXiv*, abs/1607.06450.

Alexei Baevski and Michael Auli. 2018. Adaptive input representations for neural language modeling. *arXiv*, abs/1809.10853.

Alexei Baevski, Sergey Edunov, Yinhan Liu, Luke Zettlemoyer, and Michael Auli. 2019. Cloze-driven pretraining of self-attention networks. *arXiv*.

Andrew M. Dai and Quoc V. Le. 2015. Semi-supervised sequence learning. *arXiv*, abs/1511.01432.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *CoRR*, abs/1810.04805.

Angela Fan, David Grangier, and Michael Auli. 2017. Controllable abstractive summarization. *arXiv*, abs/1711.05217.

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

Caglar Gulcehre, Orhan Firat, Kelvin Xu, Kyunghyun Cho, Loic Barrault, Huei-Chi Lin, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2015. On using monolingual corpora in neural machine translation. *arXiv*, abs/1503.03535.

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

Hakan Inan, Khashayar Khosravi, and Richard Socher. 2016. Tying word vectors and word classifiers: A loss framework for language modeling. *arXiv*, abs/1611.01462.

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

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondrej Bojar, Alexandra Constantin, and Evan Herbst. 2007. Moses: Open source toolkit for statistical machine translation. In *Proc. of ACL Demo Session*.

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

Ilya Loshchilov and Frank Hutter. 2016. SGDR: stochastic gradient descent with restarts. *arXiv*, abs/1608.03983.

Ramesh Nallapati, Bowen Zhou, Caglar Gulcehre, Bing Xiang, et al. 2016. Abstractive text summarization using sequence-to-sequence rnns and beyond. In *Proc. of CONLL*.

Myle Ott, Sergey Edunov, Alexei Baevski, Angela Fan, Sam Gross, Nathan Ng, David Grangier, and Michael Auli. 2019. fairseq: A fast, extensible toolkit for sequence modeling. In *Proc. of NAACL System Demonstrations*.

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

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. BLEU: a method for automatic evaluation of machine translation. In *Proc. of ACL*.

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

Gabriel Pereyra, George Tucker, Jan Chorowski, Lukasz Kaiser, and Geoffrey E. Hinton. 2017. Regularizing neural networks by penalizing confident output distributions. In *International Conference on Learning Representations (ICLR) Workshop*.

Matthew E Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proc. of ACL*.

Jason Phang, Thibault Fevry, and Samuel R. Bowman. 2018. Sentence encoders on stilts: Supplementary training on intermediate labeled-data tasks. *arXiv*, abs/1811.01088.

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

Ofir Press and Lior Wolf. 2017. Using the output embedding to improve language models. In *Proc. of EACL*.

Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. 2018. Improving language understanding by generative pre-training. https://s3-us-west-2.amazonaws.com/openai-assets/research-covers/language-unsupervised/language_understanding_paper.pdf.

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

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

Ilya Sutskever, James Martens, George E. Dahl, and Geoffrey E. Hinton. 2013. On the importance of initialization and momentum in deep learning. In *Proc. of ICML*.

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

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

Barret Zoph, Deniz Yuret, Jonathan May, and Kevin Knight. 2016. Transfer learning for low-resource neural machine translation. In *Proc. of EMNLP*.

A Detailed WMT English-German Results

bitext	method	**2012**	**2013**	**2014**	**2015**	**2016**	**2017**	**2018**	**Avg**
160K	baseline	13.2	15.7	13.5	15.7	18.6	14.8	21.4	16.1
	SHARED	15.3	18.2	16.7	19.0	21.6	18.2	24.9	19.1
	SHARED+bitext-BPE	15.1	17.9	16.2	18.9	22.0	18.0	25.2	19.0
	SRC-ELMO	16.0	19.4	17.1	19.9	23.0	18.7	26.6	20.1
	SRC-FT	15.3	18.5	16.6	18.9	20.8	17.6	24.3	18.9
	TGT-ELMO	13.3	16.4	14.1	16.2	18.8	14.9	21.3	16.4
	TGT-FT	14.7	17.2	15.8	18.4	21.4	16.9	24.2	18.4
	SRC-ELMO+SHDEMB	17.4	20.8	18.6	21.5	24.9	20.3	29.0	**21.8**
320K	baseline	17.2	20.4	18.1	21.2	25.0	19.6	28.9	21.5
	SHARED	18.1	21.1	19.1	22.4	26.3	21.2	30.6	22.7
	SHARED+bitext-BPE	17.6	20.6	19.1	22.3	26.1	20.8	29.9	22.3
	src-elmo	18.8	22.3	21.1	24.0	27.5	22.2	32.5	24.1
	SRC-FT	19.0	22.5	20.9	23.5	26.9	22.2	32.1	23.9
	TGT-ELMO	16.7	20.7	18.2	20.9	24.1	19.4	28.0	21.1
	TGT-FT	16.1	19.4	17.1	20.0	23.1	18.5	26.3	20.1
	SRC-ELMO+SHDEMB	19.5	22.9	21.2	24.0	27.4	22.4	32.3	**24.2**
640K	baseline	19.2	22.9	21.2	24.5	27.9	22.4	33.1	24.5
	SHARED	19.9	23.4	22.1	25.1	28.8	23.0	34.1	25.2
	SHARED+bitext-BPE	19.4	22.8	21.7	24.9	28.4	22.9	33.6	24.8
	src-elmo	21.0	24.3	23.4	26.5	30.0	24.6	35.6	26.5
	SRC-FT	20.5	24.0	22.9	26.1	29.1	24.4	34.9	26.0
	TGT-ELMO	18.9	22.6	20.8	24.2	27.5	22.3	31.9	24.0
	TGT-FT	18.2	21.8	20.6	23.7	27.0	21.8	31.4	23.5
	SRC-ELMO+SHDEMB	21.2	25.1	23.9	26.7	30.2	24.7	36.2	**26.9**
1280K	baseline	20.9	24.6	23.6	26.5	30.5	24.7	36.2	26.7
	SHARED	21.1	24.6	24.6	27.6	31.0	25.2	37.3	27.3
	SHARED+bitext-BPE	20.5	24.0	23.9	26.2	30.6	24.8	36.2	26.6
	src-elmo	22.1	25.7	25.7	28.5	31.7	26.3	38.2	28.3
	SRC-FT	21.3	25.2	25.3	28.5	31.1	26.2	37.4	27.9
	TGT-ELMO	20.9	24.4	23.6	26.6	30.3	24.9	36.1	26.7
	TGT-FT	20.1	23.7	22.4	25.2	29.1	23.6	34.4	25.5
	SRC-ELMO+SHDEMB	22.3	26.0	26.3	28.9	32.6	26.8	38.6	**28.8**
2560K	baseline	21.7	25.6	25.4	28.2	32.3	26.2	39.1	28.4
	SHARED	22.2	25.9	25.7	28.3	32.1	26.3	38.9	28.5
	SHARED+bitext-BPE	21.8	25.5	25.5	27.9	32.1	26.0	38.6	28.2
	src-elmo	22.9	27.0	27.0	30.0	33.4	28.0	40.0	**29.8**
	SRC-FT	22.2	26.4	26.3	29.5	32.4	27.3	39.3	29.1
	TGT-ELMO	21.8	25.7	25.8	28.5	32.3	26.6	39.3	28.6
	TGT-FT	21.5	25.3	24.5	27.0	30.2	25.2	36.8	27.2
	SRC-ELMO+SHDEMB	23.1	27.2	27.1	29.7	33.7	27.9	40.0	**29.8**
5186K	baseline	23.1	26.8	27.7	30.1	33.6	27.9	40.1	29.9
	SHARED	22.6	26.6	27.7	30.5	33.4	28.2	40.2	29.9
	SHARED+bitext-BPE	22.5	26.0	27.0	29.7	33.4	27.7	40.6	29.6
	src-elmo	23.7	27.8	28.7	31.1	34.5	29.2	41.8	**31.0**
	SRC-FT	23.1	27.0	27.8	30.5	33.7	28.3	40.8	30.2
	TGT-ELMO	22.9	26.6	26.9	29.5	33.8	27.7	40.5	29.7
	TGT-FT	22.3	26.1	26.1	28.9	32.5	26.5	38.8	28.7
	SRC-ELMO+SHDEMB	23.4	28.0	28.8	31.2	34.5	28.7	41.8	30.9

Table 4: BLEU on newstest2012 to newstest2018 of WMT English-German translation in varioius simulated bitext size scenarios (cf. Figure 1).

B Training and inference speed

	train (tok/sec)	inference (tok/sec)
SHARED	528,802	2,334
SRC-ELMO	100,636	2,011
SRC-FT	57,753	2,080
TGT-ELMO	142,525	259
TGT-FT	95,313	299

Table 5: Training and inference speed of models trained on WMT English-German. Training speed based on 32 V100 GPUs. Inference speed measured on a single V100 and by batching up to 12K source or target tokens.