RETURNN as a Generic Flexible Neural Toolkit with Application to Translation and Speech Recognition

Albert Zeyer[1,2,3]**, Tamer Alkhouli**[1,2] **and Hermann Ney**[1,2]

[1]Human Language Technology and Pattern Recognition Group
RWTH Aachen University, Aachen, Germany,
[2]AppTek, USA, http://www.apptek.com/,
[3]NNAISENSE, Switzerland, https://nnaisense.com/
surname@cs.rwth-aachen.de

Abstract

We compare the fast training and decoding speed of RETURNN of attention models for translation, due to fast CUDA LSTM kernels, and a fast pure TensorFlow beam search decoder. We show that a layer-wise pretraining scheme for recurrent attention models gives over 1% BLEU improvement absolute and it allows to train deeper recurrent encoder networks. Promising preliminary results on max. expected BLEU training are presented. We obtain state-of-the-art models trained on the WMT 2017 German↔English translation task. We also present end-to-end model results for speech recognition on the Switchboard task. The flexibility of RETURNN allows a fast research feedback loop to experiment with alternative architectures, and its generality allows to use it on a wide range of applications.

1 Introduction

RETURNN, the RWTH extensible training framework for universal recurrent neural networks, was introduced in (Doetsch et al., 2017). The source code is fully open[1]. It can use Theano (Theano Development Team, 2016) or TensorFlow (TensorFlow Development Team, 2015) for its computation. Since it was introduced, it got extended by comprehensive TensorFlow support. A generic recurrent layer allows for a wide range of encoder-decoder-attention or other recurrent structures. An automatic optimization logic can optimize the computation graph depending on training, scheduled sampling, sequence training, or beam search decoding. The automatic optimization together with our fast native CUDA implemented LSTM kernels allows for very fast train-

ing and decoding. We will show in speed comparisons with Sockeye (Hieber et al., 2017) that we are at least as fast or usually faster in both training and decoding. Additionally, we show in experiments that we can train very competitive models for machine translation and speech recognition. This flexibility together with the speed is the biggest strength of RETURNN.

Our focus will be on recurrent attention models. We introduce a layer-wise pretraining scheme for attention models and show its significant effect on deep recurrent encoder models. We show promising preliminary results on expected maximum BLEU training. The configuration files of all the experiments are publicly available[2].

2 Related work

Multiple frameworks exist for training attention models, most of which are focused on machine translation.

- Sockeye (Hieber et al., 2017) is a generic framework based on MXNet (Chen et al., 2015) which is most compareable to RETURNN as it is generic although we argue that RETURNN is more flexible and faster.

- OpenNMT (Levin et al., 2017a,b) based on Lua (Ierusalimschy et al., 2006) which is discontinued in development. Separate PyTorch (PyTorch Development Team, 2018) and TensorFlow implementation exists, which are more recent. We will demonstrate that RETURNN is more flexible.

- Nematus (Sennrich et al., 2017) is based on Theano (Theano Development Team, 2016) which is going to be discontinued in development. We show that RETURNN is much faster in both training and decoding as can be concluded from our speed comparison to

---

[1] https://github.com/rwth-i6/returnn

[2] https://github.com/rwth-i6/returnn-experiments/tree/master/2018-attention

Sockeye and the comparisons performed by the Sockeye authors (Hieber et al., 2017).

- Marian (Junczys-Dowmunt et al., 2016) is implemented directly in C++ for performance reasons. Again by our speed comparisons and the comparisons performed by the Sockeye authors (Hieber et al., 2017), one can conclude that RETURNN is very competitive in terms of speed, but is much more flexible.

- NeuralMonkey (Helcl and Libovický, 2017) is based on TensorFlow (TensorFlow Development Team, 2015). This framework is not as flexible as RETURNN. Also here we can conclude just as before that RETURNN is much faster in both training and decoding.

- Tensor2Tensor (Vaswani et al., 2018) is based on TensorFlow (TensorFlow Development Team, 2015). It comes with the reference implementation of the Transformer model (Vaswani et al., 2017), however, it lacks support for recurrent decoder models and overall is way less flexible than RETURNN.

3 Speed comparison

Various improved and fast CUDA LSTM kernels are available for the TensorFlow backend in RETURNN. A comparison of the speed of its own LSTM kernel vs. other TensorFlow LSTM kernels can be found on the website[3]. In addition, an automatic optimization path which moves out computation of the recurrent loop as much as possible improves the performance.

We want to compare different toolkits in training and decoding for a recurrent attention model in terms of speed on a GPU. Here, we try to maximize the batch size such that it still fits into the GPU memory of our reference GPU card, the Nvidia GTX 1080 Ti with 11 GB of memory. We keep the maximum sequence length in a batch the same, which is 60 words. We always use Adam (Kingma and Ba, 2014) for training. In Table 1, we see that RETURNN is the fastest, and also is most efficient in its memory consumption (implied by the larger batches). For these speed experiments, we did not tune any of the hyper parameters of RETURNN which explains its worse performance. The aim here is to match Sockeye's exact architecture for speed and memory comparison. During training, we observed that the learning rate scheduling settings of Sockeye are

---

[3] http://returnn.readthedocs.io/en/latest/tf_lstm_benchmark.html

toolkit	encoder n. layers	time [h]	batch size	BLEU [%] 2015	BLEU [%] 2017
RETURNN	4	**11.25**	8500	28.0	28.4
Sockeye		11.45	3000	**28.9**	**29.2**
RETURNN	6	**12.87**	7500	28.7	28.7
Sockeye		14.76	2500	**29.4**	**29.1**

Table 1: Training speed and memory consumption on WMT 2017 German→English. Train time is for seeing the full train dataset once. Batch size is in words, such that it almost maximizes the GPU memory consumption. The BLEU score is for the converged models, reported for newstest2015 (dev) and newstest2017. The encoder has one bidirectional LSTM layer and either 3 or 5 unidirectional LSTM layers.

more pessimistic, i.e. the decrease is slower and it sees the data more often until convergence. This greatly increases the total training time but in our experience also improves the model.

For decoding, we extend RETURNN with a fast pure TensorFlow beam search decoder, which supports batch decoding and can run on the GPU. A speed and memory consumption comparison is shown in Table 2. We see that RETURNN is the fastest. We report results for the batch size that yields the best speed. The slow speed of Sockeye is due to frequent cross-device communication.

toolkit	encoder n. layers	batch size [seqs]	time [secs] 2015	time [secs] 2017
RETURNN	4	50	**54**	**71**
Sockeye		5	398	581
RETURNN	6	50	**56**	**70**
Sockeye		5	403	585

Table 2: Decoding speed and memory consumption on WMT 2017 German→English. Time is for decoding the whole dataset, reported for newstest2015 (dev) and newstest2017, with beam size 12. Batch size is the number of sequences, such that it optimizes the decoding speed. This does not mean that it uses the whole GPU memory. These are the same models as in Table 1.

4 Performance comparison

We want to study what possible performance we can get with each framework on a specific task. We restrict this comparison here to recurrent attention models.

The first task is the WMT 2017 German to English translation task. We use the same 20K byte-pair encoding subword units in all toolkits (Sen-

nrich et al., 2015). We also use Adam (Kingma and Ba, 2014) in all cases. The learning rate scheduling is also similar. In RETURNN, we use a 6 layer bidirectional encoder, trained with pre-training and label smoothing. It has bidirectional LSTMs in every layer of the encoder, unlike Sockeye, which only has the first layer bidirectional. We use a variant of attention weight / fertility feedback (Tu et al., 2016), which is inverse in our case, to use a multiplication instead of a division, for better numerical stability. Our model was derived from the model presented by (Bahar et al., 2017; Peter et al., 2017) and (Bahdanau et al., 2014).

We report the best performing Sockeye model we trained, which has 1 bidirectional and 3 unidirectional encoder layers, 1 pre-attention target recurrent layer, and 1 post-attention decoder layer. We trained with a max sequence length of 75, and used the 'coverage' RNN attention type. For Sockeye, the final model is an average of the 4 best runs according to the development perplexity. The results are collected in Table 3. We obtain the best results with Sockeye using a Transformer network model (Vaswani et al., 2017), where we achieve 32.0% BLEU on newstest2017. So far, RETURNN does not support this architecture; see Section 7 for details.

toolkit	BLEU [%]	
	2015	2017
RETURNN	**31.2**	**31.3**
Sockeye	29.7	30.2

Table 3: Comparison on German→English.

We compare RETURNN to other toolkits on the WMT 2017 English→German translation task in Table 4. We observe that our toolkit outperforms all other toolkits. The best result obtained by other toolkits is using Marian (25.5% BLEU). In comparison, RETURNN achieves 26.1%. We also compare RETURNN to the best performing single systems of WMT 2017. In comparison to the fine-tuned evaluation systems that also include back-translated data, our model performs worse by only 0.3 to 0.9 BLEU. We did not run experiments with back-translated data, which can potentially boost the performance by several BLEU points.

We also have preliminary results with recurrent attention models for speech recognition on the Switchboard task, which we trained on the 300h trainset. We report on both the Switchboard (SWB) and the CallHome (CH) part of Hub5'00 and Hub5'01. We also compare to a conventional frame-wise trained hybrid deep bidirec-

System	BLEU [%] newstest2017
RETURNN	**26.1**
OpenNMT-py	21.8
OpenNMT-lua	22.6
Marian	25.6
Nematus	23.5
Sockeye	25.3
WMT 2017 Single Systems + bt data	
LMU	26.4
+ reranking	27.0
Systran	26.5
Edinburgh	26.5

Table 4: Performance comparison on WMT 2017 English→German. The baseline systems (upper half) are trained on the parallel data of the WMT Enlgish→German 2017 task. We downloaded the hypotheses from here.[4] The WMT 2017 system hypotheses (lower half) are generated using systems having additional back- translation (bt) data. These hypotheses are downloaded from here.[5]

tional LSTM with 6 layers (Zeyer et al., 2017b), and a generalized full-sum sequence trained hybrid deep bidirectional LSTM with 5 layers (Zeyer et al., 2017a). The frame-wise trained hybrid model also uses focal loss (Lin et al., 2017). All the hybrid models use a phonetic lexicon and an external 4-gram language model which was trained on the transcripts of both the Switchboard and the Fisher corpus. The attention model does not use any external language model nor a phonetic lexicon. Its output labels are byte-pair encoded subword units (Sennrich et al., 2015). It has a 6 layer bidirectional encoder, which also applies max-pooling in the time dimension, i.e. it reduces the input sequence by factor 8. Pretraining as explained in Section 6 was applied. To our knowledge, this is the best reported result for an end-to-end system on Switchboard 300h without using a language model or the lexicon. For comparison, we also selected comparable results from the literature. From these, the Baidu DeepSpeech CTC model is modeled on characters and does not use the lexicon but it does use a language model. The results are collected in Table 5.

5 Maximum expected BLEU training

We implement expected risk minimization, i.e. expected BLEU maximization or expected WER

---

model	training	WER [%]			
		Hub5'00			Hub5'01
		Σ	SWB	CH	
hybrid[1]	frame-wise		11.2		
hybrid[2]	LF-MMI	15.8	10.8		
CTC[3]	CTC	25.9	20.0	31.8	
hybrid	frame-wise	**14.4**	**9.8**	**19.0**	14.7
	full-sum	15.9	10.1	21.8	**14.5**
attention	frame-wise	20.3	13.5	27.1	19.9

Table 5: Performance comparison on Switchboard, trained on 300h. hybrid[1] is the IBM 2017 ResNet model (Saon et al., 2017). hybrid[2] trained with Lattice-free MMI (Hadian et al., 2018). CTC[3] is the Baidu 2014 DeepSpeech model (Hannun et al., 2014). Our attention model does not use any language model.

minimization, following (Prabhavalkar et al., 2017; Edunov et al., 2017). The results are still preliminary but promising. We do the approximation by beam search with beam size 4. For a 4 layer encoder network model, with forced alignment cross entropy training, we get 30.3% BLEU, and when we use maximum expected BLEU training, we get 31.1% BLEU.

6 Pretraining

RETURNN supports very generic and flexible pretraining which iteratively starts with a small model and adds new layers in the process. A similar pretraining scheme for deep bidirectional LSTMs acoustic speech models was presented earlier (Zeyer et al., 2017b). Here, we only study a layer-wise construction of the deep bidirectional LSTM encoder network of an encoder-decoder-attention model for translation on the WMT 2017 German→English task. Experimental results are presented in Table 6. The observations very clearly match our expectations, that we can both greatly improve the overall performance, and we are able to train deeper models. A minor benefit is faster training speed of the initial pretrain epochs.

encoder	BLEU [%]	
num. layers	no pretrain	with pretrain
2	29.3	-
3	29.9	-
4	29.1	30.3
5	-	30.3
6	-	30.6
7	-	**30.9**

Table 6: Pretraining comparison.

In preliminary recurrent attention experiments for speech recognition, pretraining seems very essential to get good performance.

Also, we use in all cases a learning rate scheduling scheme, which lowers the learning rate if the cross validation score does not improve enough. Without pretraining and a 2 layer encoder in the same setting as above, with a fixed learning rate, we get 28.4% BLEU, where-as with learning rate scheduling, we get 29.3% BLEU.

7 RETURNN features

Besides the fast speed, and the many features such as pretraining, scheduled sampling (Bengio et al., 2015), label smoothing (Szegedy et al., 2016), and the ability to train state-of-the-art models, one of the greatest strengths of RETURNN is its flexibility. The definition of the recurrent dependencies and the whole model architecture are provided in a very explicit way via a config file. Thus, e.g. trying out a new kind of attention scheme, adding a new latent variable to the search space, or drastically changing the whole architecture, is all supported already and does not need any more implementation in RETURNN. All that can be expressed by the neural network definition in the config. A (simplified) example of a network definition is given in Listing 1.

Each layer in this definition does some computation, specified via the `class` attribute, and gets its input from other layers via the `from` attribute, or from the input data, in case of layer `src`. The `output` layer defines a whole subnetwork, which can make use of recurrent dependencies via a `prev:` prefix. Depending on whether training or decoding is done, the `choice` layer class would return the true labels or the predicted labels. In case of scheduled sampling or max BLEU training, we can also use the predicted label during training. Depending on this configuration, during compilation of the computation graph, RETURNN figures out that certain calculations can be moved out of the recurrent loop. This automatic optimization also adds to the speedup. This flexibility and ease of trying out new architectures and models allow for a very efficient development / research feedback loop. Fast, consistent and robust feedback greatly helps the productivity and quality. This is very different to other toolkits which only support a predefined set of architectures.

To summarize the features of RETURNN:
- flexibility (see above),
- generality, wide range of models and appli-

```
network = {
# recurrent bidirectional encoder:
"src": {"class": "linear", "n_out": 620}, # embedding
"enc0_fw": {"class": "rec", "unit": "nativelstm2", "n_out": 1000, "direction": 1, "from": ["src"]},
"enc0_bw": {"class": "rec", "unit": "nativelstm2", "n_out": 1000, "direction": -1, "from": ["src"]},
# ... more encoder LSTM layers

"encoder": {"class": "copy", "from": ["enc5_fw", "enc5_bw"]},
"enc_ctx": {"class": "linear", "from": ["encoder"], "n_out": 1000},

# recurrent decoder:
"output": {"class": "rec", "from": [], "unit": {
  "output": {"class": "choice", "from": ["output_prob"]},
  "trg": {"class": "linear", "from": ["output"], "n_out": 620, "initial_output": 0},
  "weight_feedback": {"class": "linear", "from": ["prev:accum_a"], "n_out": 1000},
  "s_tr": {"class": "linear", "from": ["s"], "n_out": 1000},
  "e_in": {"class": "combine", "kind": "add", "from": ["base:enc_ctx", "weight_feedback", "s_tr"]},
  "e_tanh": {"class": "activation", "activation": "tanh", "from": ["e_in"]},
  "e": {"class": "linear", "from": ["e_tanh"], "n_out": 1},
  "a": {"class": "softmax_over_spatial", "from": ["e"]},
  "accum_a": {"class": "combine", "kind": "add", "from": ["prev:accum_a", "a"]},
  "att": {"class": "generic_attention", "weights": "a", "base": "base:encoder"},
  "s": {"class": "rnn_cell", "unit": "LSTMBlock", "from": ["prev:trg", "prev:att"], "n_out": 1000},
  "readout": {"class": "linear", "activation": "relu", "from": ["s", "prev:trg", "att"], "n_out": 1000},
  "output_prob": {"class": "softmax", "from": ["readout"], "dropout": 0.3, "loss": "ce",
                  "loss_opts": {"label_smoothing": 0.1}}
}},
"decision": {"class": "decide", "from": ["output"], "loss": "bleu"}
}
```

Listing 1: RETURNN config example for an attention model

cations, such as hybrid acoustic speech models, language models and attention models for translation and speech recognition,

- fast CUDA LSTM kernels,
- attention models, generic recurrent layer, fast beam search decoder,
- sequence training (min WER, max BLEU),
- label smoothing, scheduled sampling,
- TensorFlow backend and the old Theano backend, which has a separate fast attention implementation (Doetsch et al., 2016), fast CUDA MDLSTM kernels (Voigtlaender et al., 2016), as well as fast sequence training (Zeyer et al., 2017c).

One feature which is currently work-in-progress is the support for self-attention in the recurrent layer. The reason this needs some more work is because we currently only support access to the previous time step (`prev:`) but not to the whole past, which is needed for self-attention. That is why we did not present any Transformer (Vaswani et al., 2017) comparisons yet.

8 Conclusion

We have demonstrated many promising features of RETURNN and presented state-of-the-art systems in translation and speech recognition. We argue that it is a convenient testbed for research and applications. We introduced pretraining for recurrent attention models and showed its advantages while not having any disadvantages. Maximum expected BLEU training seems to be promising.

References

Parnia Bahar, Jan Rosendahl, Nick Rossenbach, and Hermann Ney. 2017. The RWTH Aachen machine translation systems for IWSLT 2017. In *IWSLT*, pages 29–34, Tokyo, Japan.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

Samy Bengio, Oriol Vinyals, Navdeep Jaitly, and Noam Shazeer. 2015. Scheduled sampling for sequence prediction with recurrent neural networks. In *NIPS*, pages 1171–1179.

Tianqi Chen, Mu Li, Yutian Li, Min Lin, Naiyan Wang, Minjie Wang, Tianjun Xiao, Bing Xu, Chiyuan Zhang, and Zheng Zhang. 2015. Mxnet: A flexible and effi-

cient machine learning library for heterogeneous distributed systems. *arXiv preprint arXiv:1512.01274*.

Patrick Doetsch, Albert Zeyer, and Hermann Ney. 2016. Bidirectional decoder networks for attention-based end-to-end offline handwriting recognition. In *ICFHR*, pages 361–366, Shenzhen, China.

Patrick Doetsch, Albert Zeyer, Paul Voigtlaender, Ilia Kulikov, Ralf Schlüter, and Hermann Ney. 2017. RE-TURNN: the RWTH extensible training framework for universal recurrent neural networks. In *ICASSP*, pages 5345–5349, New Orleans, LA, USA.

Sergey Edunov, Myle Ott, Michael Auli, David Grangier, and Marc'Aurelio Ranzato. 2017. Classical structured prediction losses for sequence to sequence learning. *arXiv preprint arXiv:1711.04956*.

Hossein Hadian, Hossein Sameti, Daniel Povey, and Sanjeev Khudanpur. 2018. Towards discriminatively-trained HMM-based end-to-end model for automatic speech recognition. Submitted to ICASSP 2018.

Awni Hannun, Carl Case, Jared Casper, Bryan Catanzaro, Greg Diamos, Erich Elsen, Ryan Prenger, Sanjeev Satheesh, Shubho Sengupta, Adam Coates, et al. 2014. DeepSpeech: Scaling up end-to-end speech recognition. *arXiv preprint arXiv:1412.5567*.

Jindřich Helcl and Jindřich Libovický. 2017. Neural monkey: An open-source tool for sequence learning. *The Prague Bulletin of Mathematical Linguistics*, 107(1):5–17.

Felix Hieber, Tobias Domhan, Michael Denkowski, David Vilar, Artem Sokolov, Ann Clifton, and Matt Post. 2017. Sockeye: A toolkit for neural machine translation. *arXiv preprint arXiv:1712.05690*.

Roberto Ierusalimschy, Luiz Henrique de Figueiredo, and Waldemar Celes. 2006. *Lua 5.1 Reference Manual*. Lua.Org.

Marcin Junczys-Dowmunt, Tomasz Dwojak, and Hieu Hoang. 2016. Is neural machine translation ready for deployment? a case study on 30 translation directions. *arXiv preprint arXiv:1610.01108*.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Pavel Levin, Nishikant Dhanuka, Talaat Khalil, Fedor Kovalev, and Maxim Khalilov. 2017a. Toward a full-scale neural machine translation in production: the booking. com use case. *MT Summit, arXiv preprint arXiv:1709.05820*.

Pavel Levin, Nishikant Dhanuka, and Maxim Khalilov. 2017b. Machine translation at booking. com: Journey and lessons learned. *arXiv preprint arXiv:1707.07911*.

Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. 2017. Focal loss for dense object detection. *arXiv preprint arXiv:1708.02002*.

Jan-Thorsten Peter, Andreas Guta, Tamer Alkhouli, Parnia Bahar, Jan Rosendahl, Nick Rossenbach, Miguel Graça, and Ney Hermann. 2017. The RWTH Aachen university english-german and german-english machine translation system for WMT 2017. In *EMNLP*, Copenhagen, Denmark.

Rohit Prabhavalkar, Tara N Sainath, Yonghui Wu, Patrick Nguyen, Zhifeng Chen, Chung-Cheng Chiu, and Anjuli Kannan. 2017. Minimum word error rate training for attention-based sequence-to-sequence models. *arXiv preprint arXiv:1712.01818*.

PyTorch Development Team. 2018. PyTorch. Software available from pytorch.org.

George Saon, Gakuto Kurata, Tom Sercu, Kartik Audhkhasi, Samuel Thomas, Dimitrios Dimitriadis, Xiaodong Cui, Bhuvana Ramabhadran, Michael Picheny, Lynn-Li Lim, et al. 2017. English conversational telephone speech recognition by humans and machines. *arXiv preprint arXiv:1703.02136*.

Rico Sennrich, Orhan Firat, Kyunghyun Cho, Alexandra Birch, Barry Haddow, Julian Hitschler, Marcin Junczys-Dowmunt, Samuel Läubli, Antonio Valerio Miceli Barone, Jozef Mokry, et al. 2017. Nematus: a toolkit for neural machine translation. *EACL Demo, arXiv preprint arXiv:1703.04357*.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2015. Neural machine translation of rare words with subword units. *arXiv preprint arXiv:1508.07909*.

Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. 2016. Rethinking the inception architecture for computer vision. In *IC-CVPR*, pages 2818–2826.

TensorFlow Development Team. 2015. TensorFlow: Large-scale machine learning on heterogeneous systems. Software available from tensorflow.org.

Theano Development Team. 2016. Theano: A Python framework for fast computation of mathematical expressions. *arXiv e-prints*, abs/1605.02688.

Zhaopeng Tu, Zhengdong Lu, Yang Liu, Xiaohua Liu, and Hang Li. 2016. Modeling coverage for neural machine translation. In *ACL*.

Ashish Vaswani, Samy Bengio, Eugene Brevdo, Francois Chollet, Aidan N Gomez, Stephan Gouws, Llion Jones, Łukasz Kaiser, Nal Kalchbrenner, Niki Parmar, et al. 2018. Tensor2tensor for neural machine translation. *arXiv preprint arXiv:1803.07416*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *NIPS*, pages 6000–6010.

Paul Voigtlaender, Patrick Doetsch, and Hermann Ney. 2016. Handwriting recognition with large multidimensional long short-term memory recurrent neural networks. In *ICFHR*, pages 228–233, Shenzhen, China. IAPR Best Student Paper Award.

Albert Zeyer, Eugen Beck, Ralf Schlüter, and Hermann Ney. 2017a. CTC in the context of generalized full-sum HMM training. In *Interspeech*, pages 944–948, Stockholm, Sweden.

Albert Zeyer, Patrick Doetsch, Paul Voigtlaender, Ralf Schlüter, and Hermann Ney. 2017b. A comprehensive study of deep bidirectional LSTM RNNs for acoustic modeling in speech recognition. In *ICASSP*, pages 2462–2466, New Orleans, LA, USA.

Albert Zeyer, Ilia Kulikov, Ralf Schlüter, and Hermann Ney. 2017c. Faster sequence training. In *ICASSP*, pages 5285–5289, New Orleans, LA, USA.