Parameter Optimization for CTC Acoustic Models in a Less-resourced Scenario: An Empirical Study

Markus Müller, Sebastian Stüker, Alex Waibel

Institute for Anthropomatics and Robotics, Karlsruhe Institute of Technology, Karlsruhe, Germany
Email: `{m.mueller,sebastian.stueker,alexander.waibel}@kit.edu`

Abstract

In this work, we performed a study evaluating parameter configurations for acoustic models trained using the connectionist temporal classification (CTC) loss function. We varied the sizes of the hidden layers and mini-batches. We further used different optimizer strategies, comparing using SGD with a static learning rate and newbob scheduling. We further evaluated multiple configurations for data augmentation: As the acoustic features were extracted using overlapping windows, neighboring frames contain redundant information. By systematically omitting frames, thus creating multiple versions of the same utterances, the network can be trained on "more" data. Applying this scheme, we generated multiple versions of each utterance using different shifts.

We evaluated parameter configurations using two conditions: A multilingual setup with 4 languages and 45h of data per language which can be considered less-resourced in the regimen of RNN/CTC acoustic models. In addition, we used a very low-resource dataset to assess network configurations if less than 10h of data is available.

1 Introduction

All-neural approaches in speech recognition are becoming increasingly popular. Using the connectionist temporal classification (CTC) loss function [1], such systems are trained on sequences of output tokens directly, without the need of traditional HMM-based systems for bootstrapping. CTC based systems typically make use of recurrent neural networks (RNNs) like bi-directional LSTM (BiLSTM) based ones, which are powerful tools for sequence modelling. With less explicitly modeled components, the amount of data used during training has a larger influence on the system performance because the neural networks have to learn implicitly what is modeled explicitly in a traditional system.

In this work, we evaluated multiple parameter configurations to determine the best system setup. Related studies have been published in the past, covering the aspects of how the utterances are shuffled during training [2] or BiLSTMs in general [3].

This paper is organized as follows: in the next Section 2, we provide a brief overview of multilingual RNN/CTC based setups. In Section 3, we outline the parameters analyzed in this study. A description of our experimental setup is given in Section 4, followed by the results in Section 5. This paper concludes in Section 6, where we also provide an outlook to future work.

2 Related Work

Systems trained multilingually using all-neural architectures were proposed in the past. To the best of our knowledge, we were the first to propose using a global set of acoustic units [4] with a CTC/RNN setup, which we then improved in multiple iterations [5–7]. A similar approach was proposed [8], but with the difference of providing the language identity explicitly. There was also an approach [9] for this task proposed using and end-to-end architecture with an attention mechanism [10]. Using attention in combination with CTC was proposed [11] and refined [12] for better code switching.

3 Parameter Optimization

There are multiple design decisions in building a neural network based system. In this work, we evaluated using 1) mini-batches of different sizes, 2) hidden layers of different sizes, 3) data augmentation, 4) different optimizers and 5) dropout.

3.1 Mini-batch Size

We evaluted the use of different mini-batch sizes. In order to account for the increased amount of GPU memory needed for larger mini-batches, we used smaller mini-batches and accumulated the gradients of several mini-batches and applied updates to the parameters not after every mini-batch. Instead of e.g. using a mini-batch size of 15, we used a mini-batch size of 5 and accumulated the updates of 3 mini-batches and applied them only after the group of 3 mini-batches. While this increased the computation time, it made the network training more memory efficient.

3.2 Hidden Layer Size

Using this method for saving GPU memory, we were also able to train networks with larger hidden layers. While we were previously only able to train this network architecture using at most 420 BiLSTM cells per layer, we are now able use up to 1260 BiLSTM cells per layer.

3.3 Data Augmentation

Speech is a highly variable signal. In order to increase the robustness of our setup, we used a data augmentation technique similar to the one used in EESEN [13]. We created multiple versions of each utterance by stacking feature frames and omitting every n-th frame. As acoustic input features, we were using multilingual bottleneck features (ML-BNFs). These features are computed based on the logMel scaled spectrogram. To extract the logMels, we used a windowsize of 32ms and a frameshift of 10ms. Adjacent logMel feature frame therefore already contain redundancies. As input for the ML-BNFs, a context of $+/-$ 6 frames was used, which again resulted in overlap between adjacent ML-BNF frames.

As input to our main network, we further stacked the ML-BNF frames, adding a context of 1 frame towards each side, hence the network is input a stack of 3 frames for each time step. Because of the redundant information, we evaluated using only very n-th frame. By augmenting the data with e.g. factor 2, we would omit every second frame, but due to the context the network would still see every frame. To create multiple versions of the same utterance, we omitted frames at the beginning of each utterance. When generating e.g. 4 versions of each utterance, we would omit in term 0, 1, 2 or 3 frames at the beginning. This results in a shift in contexts and therefore creates a slight variation between the different versions of each utterance. In addition, we would omit every n-th frame after applying the context.

4 Experimental setup

For neural network training, we used CTC.ISL[1], a toolkit we developed to train RNNs using the CTC loss function. For feature extraction, we used the acoustic front-end of the Janus Recognition Toolkit (JRTk) [14] which features the IBIS decoder [15]. We used the Euronews corpus [16] for our experiments. It contains recordings of the Euronews TV broadcast station covering multiple languages. In this work, we used English, French, German and Turkish data from this corpus for system training. The two conditions evaluated are either a system trained on 4 languages using 45h per language, or an additional very-low resource subset containing only 2h per language.

As input features of our system, we used multilingual bottleneck features (ML-BNFs). This bottleneck network was trained on Euronews data using a combination of 5 languages (French, German, Italian, Russian, Turkish), explicitly omitting English from the training data which serves as contrasting language to evaluate the performance on a language not seen during ML-BNF training.

4.1 Baseline Setup

As baseline setup, we used a network with 4 BiLSTM layers and 420 BiLSTM cells per layer. The network is divided into two blocks of 2 layers. The network architecture for our multilingual setup is shown in Figure 1. The language adaptation [6] is applied between the two BiLSTM blocks. The networks were trained using graphemes. No pronunciation dictionary was used, hence the networks needed to learn pronunciation rules by themselves. A special token was used indicating word boundaries. This forced the networks to learn a basic language model, as word boundaries typically do not have an acoustic representation.

4.2 Evaluation

To evaluate the recognition accuracy of the networks we computed the character-error-rate (CER) after applying a greedy-decoding [1]. No external models, e.g. language models, were used. As the system is trained on single characters, it is an open vocabulary system.

4.3 Conditions

We evaluated optimizing each parameter in turn, starting with the size of the hidden layers. Next, we evaluated using different factors of data augmentation, starting with very little data in a very-low resource condition. We then used the best condition, to re-train networks on more data as contrasting experiment. In addition, we evaluated different optimizers. Starting with SGD with Nesterov momentum as baseline, we evaluated using newbob scheduling. In a final experiment, we applied dropout to our networks. In cases where the very-low resource subset was used, we reduced the number of BiLSTM cells per layer to 210 to avoid overfitting.

[1]https://github.com/markus-m-u-e-l-l-e-r/CTC.ISL

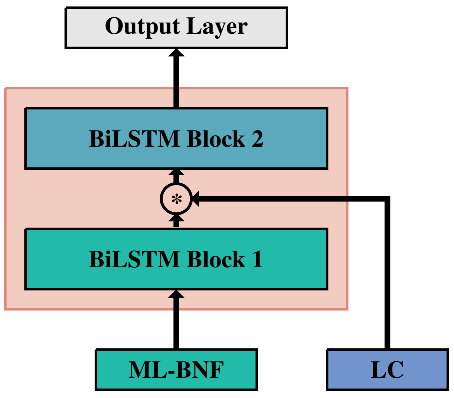

Figure 1: Network architecture used, showing the adaptation with a language code (LC) during multilingual training.

5 Results

We provide the results for each language in our training set individually, as well as averaged over all 4 languages (AVG).

5.1 Network Size

We evaluated using BiLSTM hidden layers with 420, 840 and 1,260 BiLSTM cells. As shown in Table 1, increasing the size of the layers also improves the performance. The lowest CERs were observed by using 1,260 BiLSTM cells per layer. Figure 2 shows a graph of the CERs over the epochs. We did not evaluate increasing the number of cells beyond this amount because of increased computational costs and limitations in GPU memory.

Layer Size	AVG	DE	EN	FR	TR
420	7.9	6.0	12.1	9.1	5.5
840	6.8	5.0	10.3	8.1	4.7
1260	**6.4**	**4.7**	**9.7**	**7.6**	**4.4**

Table 1: Hidden layer sizes, showing CERs.

5.2 Dropout

We choose the two best network configurations (840 and 1,260 BiLSTM cells per layer) from the previous experiments and applied dropout training with a dropout factor of 0.2. The results are shown in Tables 2 for 840 BiLSTM cells per layer and in Table 3 using 1,260 BiLSTM cells per layer. Figure 3 shows as the CERs decrease during training. Applying dropout decreases the CER in both cases, although the improvements are only marginal.

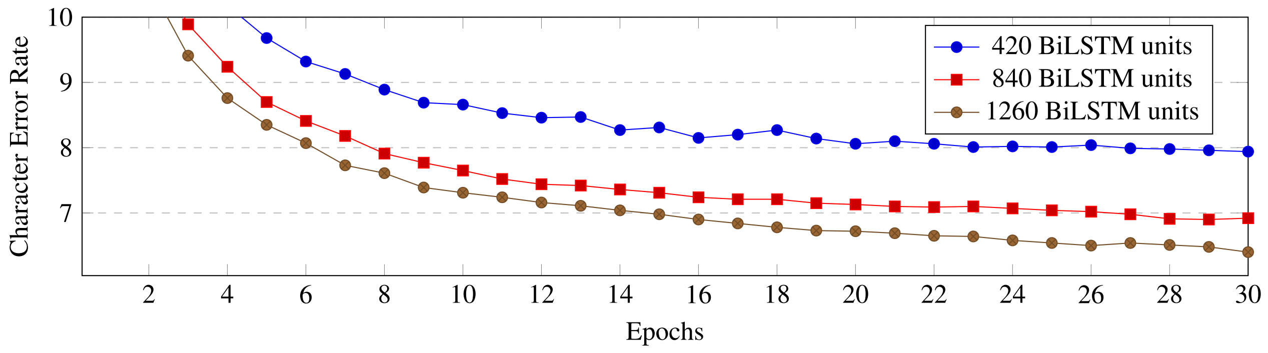

Figure 2: Comparison of hidden layer sizes, showing CERs averaged over all 4 languages.

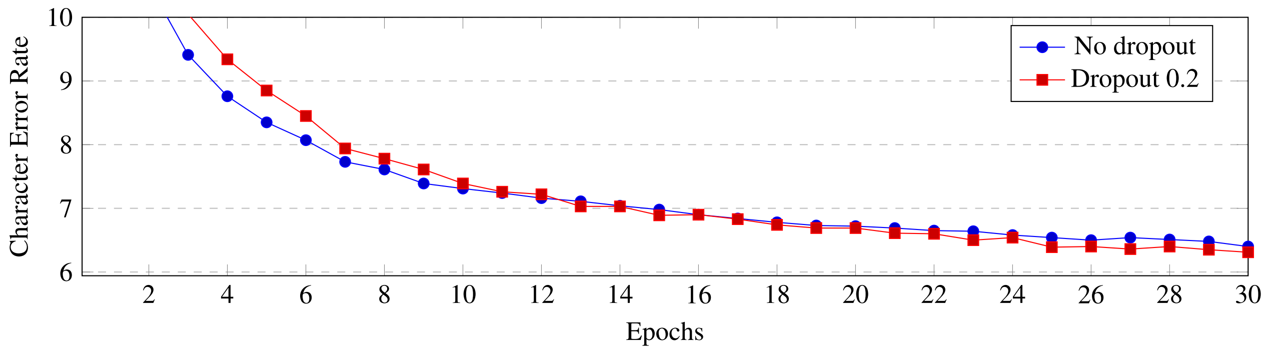

Figure 3: Comparison of dropout training, using 1,260 BiLSTM cells, showing CERs averaged over all 4 languages.

Dropout	AVG	DE	EN	FR	TR
0.0	6.8	5.0	10.3	8.1	4.7
0.2	**6.5**	**4.8**	**9.8**	**7.6**	**4.7**

Table 2: Applying dropout, 840 BiLSTM cells, showing CERs.

Dropout	AVG	DE	EN	FR	TR
0.0	6.4	4.7	9.7	7.6	4.4
0.2	**6.3**	**4.5**	**9.5**	**7.5**	**4.3**

Table 3: Applying dropout, 1,260 BiLSTM cells, showing CERs.

5.3 Optimizers

For the evaluation of different optimizers, we used the very low-resource subset of the data because of the reduced training time. To account for less available data, we reduced the number of BiLSTM cells per layer to 210 and applied dropout training with a factor of 0.2. The results are shown in Table 4 and Figure 4. Using SGD and Adam resulted in similar performance, but using SGD with newbob scheduling decreased the CERs.

5.4 Data Augmentation

We evaluated using data augmentation in two conditions. First we used the very low-resource subset of the data to evaluate multiple augmentation factors. The results are shown in Table 5 and Figure 5. Based on the best configuration (augmentation factor 2), we trained networks using the full dataset. The results as shown in Table 6 and Figure 6 indicate similar improvements to using the very low-resource subset.

Optimizer	AVG	DE	EN	FR	TR
SGD	27.4	21.2	35.9	32.5	22.2
Adam	27.4	21.6	37.3	32.6	20.5
SGD+NB	**25.0**	**19.0**	**34.3**	**31.1**	**18.2**

Table 4: Comparison of optimizers, very-low resource condition, showing CERs.

Augmentation	AVG	DE	EN	FR	TR
None	23.6	18.0	32.6	29.6	16.5
Factor 2	**23.1**	**17.3**	**31.8**	**28.8**	**16.7**
Factor 3	25.0	19.0	34.3	31.1	18.2
Factor 4	27.7	21.6	37.4	33.9	20.2

Table 5: Augmentation factors in very-low resource condition, showing CERs.

Augmentation	AVG	DE	EN	FR	TR
None	6.9	5.1	10.3	8.2	4.8
Factor 2	**6.6**	**4.9**	**10.1**	**7.6**	**4.6**

Table 6: Augmentation factors using the full dataset, showing CERs.

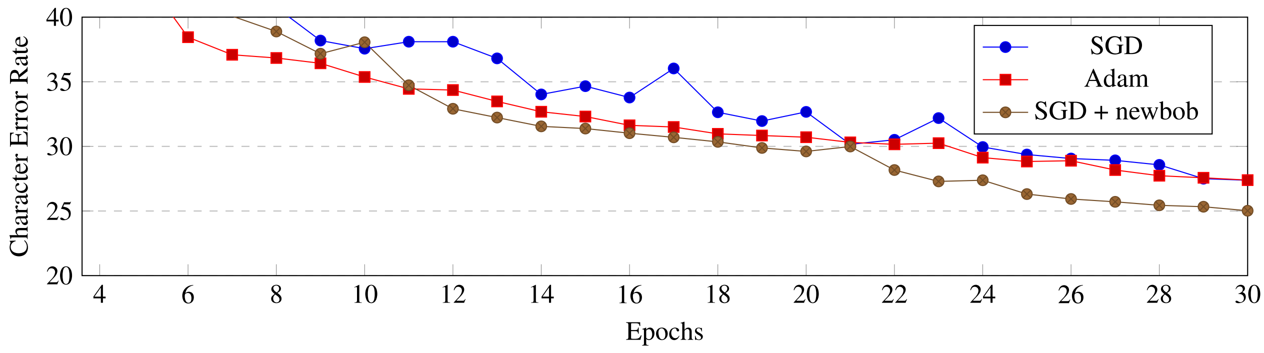

Figure 4: Comparison of different optimizer strategies, using the very-low resource subset, showing CERs averaged over all 4 languages.

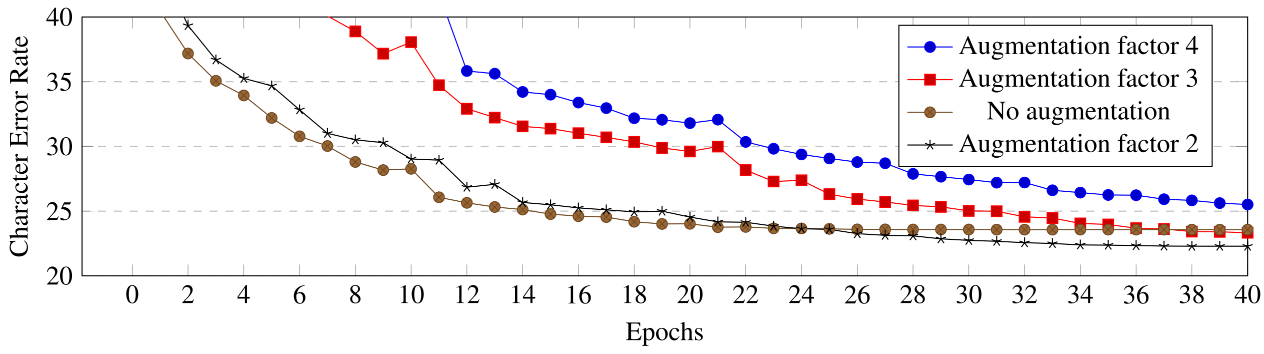

Figure 5: Comparison of data augmentation factors in very-low resource condition, showing CERs averaged over all 4 languages.

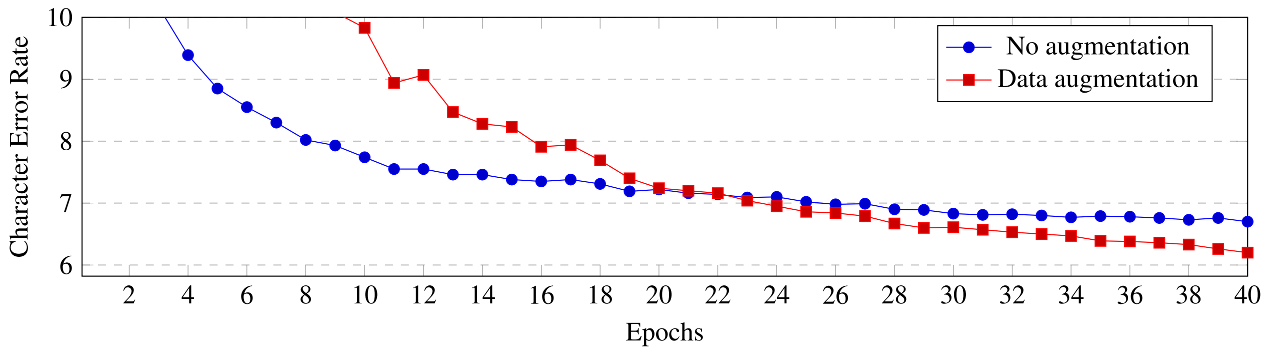

Figure 6: Data augmentation using the full dataset, showing CERs averaged over all 4 languages.

5.5 Mini-batch sizes

As final evaluation, we increased the size of the mini-batches from 15 to 60 utterances. Using smaller mini-batches results in more localized updates which may contain more noise, whereas larger mini-batches result in smoother updates which may be to generic. As the results in Table 7 indicate, using a mini-batch size of 60 decreases the performance.

Mini-batch Size	AVG	DE	EN	FR	TR
15	**7.9**	**6.0**	**12.1**	**9.1**	**5.5**
60	8.1	6.0	12.1	9.4	5.6

Table 7: Mini-batch sizes, showing CERs.

6 Conclusion

We have shown the effects of multiple parameters on the system performance. By using delayed weight updates with smaller mini-batches, we were able to increase the size of the hidden layers in our network without increasing the amount of necessary GPU memory. This enabled both the use of larger networks and mini-batches.

Based on our experiments, using a network architecture with 1260 BiLSTM cells per layer, SGD with newbob scheduling, data augmentation with factor 2 and dropout training with a factor of 0.2 resulted in the best performance for the evaluated dataset. Future work includes additional experiments to further optimize the mini-batch size and the application of dropout training which we only addressed briefly in this work.

References

[1] A. Graves, S. Fernández, F. Gomez, and J. Schmidhuber, "Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks," in *Proceedings of the 23rd international conference on Machine learning*, pp. 369–376, ACM, 2006.

[2] P. Doetsch, P. Golik, and H. Ney, "A comprehensive study of batch construction strategies for recurrent neural networks in mxnet," *arXiv preprint arXiv:1705.02414*, 2017.

[3] A. Zeyer, P. Doetsch, P. Voigtlaender, R. Schlüter, and H. Ney, "A Comprehensive Study of Deep Bidirectional LSTM RNNs for Acoustic Modeling in Speech Recognition," *arXiv preprint arXiv:1606.06871*, 2016.

[4] M. Müller, S. Stüker, and A. Waibel, "Language adaptive multilingual CTC speech recognition," in *International Conference on Speech and Computer*, pp. 473–482, Springer, 2017.

[5] M. Müller, S. Stüker, and A. Waibel, "Phonemic and graphemic multilingual ctc based speech recognition," *arXiv preprint arXiv:1711.04564*, 2017.

[6] M. Müller, S. Stüker, and A. Waibel, "Multilingual adaptation of rnn based asr systems," in *Acoustics, Speech and Signal Processing (ICASSP), 2018 IEEE International Conference on*, IEEE, 2018.

[7] M. Müller, S. Stüker, and A. Waibel, "Neural language codes for multilingual acoustic models," in *Interspeech*, 2018. submitted to.

[8] S. Kim and M. L. Seltzer, "Towards language-universal end-to-end speech recognition," *arXiv preprint arXiv:1711.02207*, 2017.

[9] S. Toshniwal, T. N. Sainath, R. J. Weiss, B. Li, P. Moreno, E. Weinstein, and K. Rao, "Multilingual speech recognition with a single end-to-end model," *arXiv preprint arXiv:1711.01694*, 2017.

[10] W. Chan, N. Jaitly, Q. Le, and O. Vinyals, "Listen, attend and spell: A neural network for large vocabulary conversational speech recognition," in *Acoustics, Speech and Signal Processing (ICASSP), 2016 IEEE International Conference on*, pp. 4960–4964, IEEE, 2016.

[11] S. Watanabe, T. Hori, and J. R. Hershey, "Language independent end-to-end architecture for joint language identification and speech recognition," in *Automatic Speech Recognition and Understanding Workshop (ASRU), 2017 IEEE*, pp. 265–271, IEEE, 2017.

[12] H. Seki, S. Watanabe, T. Hori, J. Le Roux, and J. Hershey, "An end-to-end language-tracking speech recognizer for mixed-language speech," 2018.

[13] Y. Miao, M. Gowayyed, and F. Metze, "EESEN: End-to-end Speech Recognition Using Deep RNN Models and WFST-based Decoding," in *Automatic Speech Recognition and Understanding (ASRU), 2015 IEEE Workshop on*, pp. 167–174, IEEE, 2015.

[14] M. W. et al., "JANUS 93: Towards Spontaneous Speech Translation," in *International Conference on Acoustics, Speech, and Signal Processing 1994*, (Adelaide, Australia), 1994.

[15] H. Soltau, F. Metze, C. Fugen, and A. Waibel, "A one-pass decoder based on polymorphic linguistic context assignment," in *Automatic Speech Recognition and Understanding, 2001. ASRU'01. IEEE Workshop on*, pp. 214–217, IEEE, 2001.

[16] R. Gretter, "Euronews: A Multilingual Benchmark for ASR and LID," in *Fifteenth Annual Conference of the International Speech Communication Association*, 2014.