IMPLEMENTING EASY-TO-USE RECIPES FOR THE SWITCHBOARD BENCHMARK

Dominik Wagner[1], *Sebastian P. Bayerl*[1], *Tobias Bocklet*[1]

[1]*Technische Hochschule Nürnberg Georg Simon Ohm*
dominik.wagner@th-nuernberg.de

Abstract: We report on our contribution of templates for tokenization, language modeling, and automatic speech recognition (ASR) on the Switchboard benchmark to the open-source general-purpose toolkit SpeechBrain. Three recipes for the training of end-to-end ASR systems were implemented. We describe their model architectures, as well as the necessary data preparation steps. The word error rates achievable with our models are comparable to or better than those of other popular toolkits. Pre-trained ASR models were made available on HuggingFace. They can be easily integrated into research projects or used directly for quick inference via a hosted inference API.

1 Introduction

Since its inception, the SpeechBrain toolkit [1] has made significant progress as an open-source general-purpose toolkit for speech processing tasks. Its focus lies on the provision of state-of-the-art templates (recipes) for a variety of speech-related tasks, such as automatic speech recognition (ASR), speaker identification, and speech enhancement. The toolkit is designed to be easy to use by speech researchers and the broader machine learning community alike. It allows for rapid prototyping of new deep learning methods and its codebase aims to be easily understandable for a large user base. SpeechBrain is developed on top of PyTorch [2]. Unlike other toolkits, it has a minimal list of external dependencies that are all available via the Python Package Index (PyPI). No programming languages other than Python are used to implement data processing, model training, and the model architectures themselves.
The ASR parts of the toolkit focus on models for end-to-end speech recognition, which directly map acoustic features to a word sequence. Popular end-to-end architectures based on transformers [3] and wav2vec 2.0 (W2V2) [4] have recently outperformed the conventional hybrid HMM-DNN framework for speech recognition.
SpeechBrain provides several templates for speech recognition benchmarks, such as LibriSpeech and CommonVoice. However, no recipes for the widely used Switchboard corpus [5] have been included so far. Switchboard remains an important benchmark for speech recognition tasks among researchers and exists in other popular speech recognition toolkits such as ESPNet [6] and Kaldi [7].
In this work, we report on our contribution of Switchboard recipes for tokenization, language modeling, and speech recognition to the SpeechBrain project. We describe the system architectures, as well as the necessary data preparation steps. We discuss experimental results and compare them to other toolkits. Finally, we show how pre-trained Switchboard models can be easily integrated into other projects.

2 Method

We implement one recipe for tokenizer training, two recipes for language model (LM) training, and three recipes for the training of ASR systems. The tokenizer recipe allows for the training of SentencePiece [8] models on the corpus transcripts. The recipe is configured to generate 1000 subword units estimated via unigram language modeling as recognition tokens. We offer two recipes for language modeling. The first recipe is designed for finetuning a transformer [3] language model pre-trained on the full LibriSpeech dataset (10 million words) [9] on the Switchboard and Fisher transcripts. The second recipe allows for the training of a transformer LM on the Switchboard and Fisher transcripts from scratch.
The three end-to-end ASR recipes include a CNN-RNN system, a transformer model, and a model that uses wav2vec 2.0 [4] (W2V2) to encode speech features. The CNN-RNN system consists of a tokenizer trained on the training transcripts of the Switchboard and the Fisher corpus, as well as an acoustic model (AM). The AM is made of blocks of convolutional neural networks (CNN) with normalization and pooling in the frequency domain, followed by a bidirectional long short-term memory network that is connected to a final feedforward neural network to obtain the acoustic representation, that is passed to a hybrid connectionist temporal classification (CTC) and attention decoder [10].
The transformer system consists of three blocks: a tokenizer, a transformer language model, and an acoustic model made of a transformer encoder as well as a joint CTC/transformer decoder.
The W2V2-based system consists only of a tokenizer and an acoustic model component. A pre-trained W2V2 model (`facebook/wav2vec2-large-lv60`) is combined with an encoder consisting of three linear layers. The final acoustic representation generated by the model is passed to a greedy CTC decoder.
The block diagrams in Figure 1 illustrate the composition of the three ASR architectures. Unless otherwise stated, we use 80-dimensional Mel filterbank features with a frame-width of 25ms and a frame-shift of 10ms as acoustic input features. Our ASR recipes employ 1000 subword units estimated via unigram language modeling as recognition tokens [11]. Each of these default parameters can be adapted to individual needs with SpeechBrain's YAML-based configuration system.

2.1 Data

The Switchboard benchmark is comprised of a main corpus for acoustic model training, as well as two additional corpora for performance evaluation and to supplement the amount of training data for language modeling.
The main corpus used for acoustic model training is the Switchboard-1 Telephone Speech Corpus (Swbd) [5]. It consists of approximately 260 hours of English telephone conversations. The audio files are two channel interleaved mulaw in sphere format sampled at 8 kHz. The Swbd corpus contains approximately 2,400 two-sided telephone conversations among 543 speakers (302 male, 241 female) from the United States. The speech from the two subjects is recorded on separate channels. The conversations cover a wide range of topics and no two speakers conversed together more than once.
The 2000 HUB5 English Evaluation (Eval2000) corpus [12] is used for performance measurement. It consists of approximately 11 hours of English conversational telephone speech. The corpus is divided into two parts: 20 unreleased telephone conversations from the Switchboard studies and 20 telephone conversations from CALLHOME American English Speech [13], which comprises unscripted telephone conversations between native English speakers.
The transcripts of part 1 and part 2 of the Fisher English Training Speech (Fisher) [14, 15]

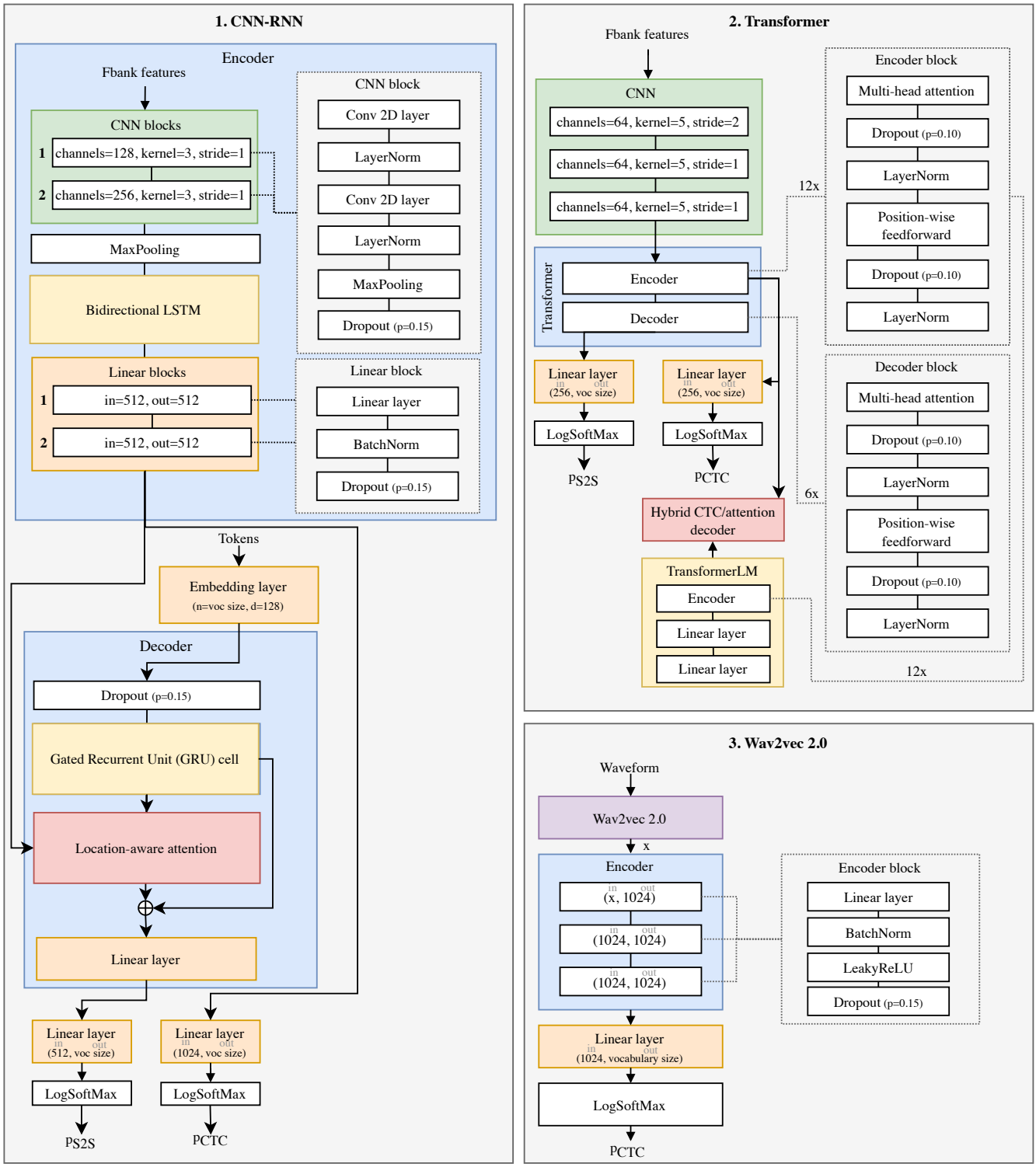

Figure 1 – Model architectures of the three ASR recipes.

can be used to provide more training data for language models and tokenizers. Both parts of the Fisher corpus contain time-aligned transcripts for 11,699 recorded telephone conversations, totaling approximately 1,960 hours of speech.

2.2 Data Preparation

We follow the data preparation steps provided by the `swbd/s5c` recipe of the Kaldi toolkit. This has several advantages: the pipeline is well-tested, and we ensure the best possible comparability to other toolkits, that also rely on the same data processing steps (e.g. ESPNet and

Kaldi). However, the integration of all data preparation steps poses a challenge due to the Python-centric approach chosen by the developers of the SpeechBrain toolkit. The major portion of Kaldi's data preparation steps has been implemented in Bash and Perl scripts, and therefore needs to be rewritten in Python. We implemented all necessary steps in a single script (`switchboard_prepare.py`) that expects valid path specifiers to the corpora described in subsection 2.1 and returns CSV files containing information such as duration, begin, end, channel, path to the audio file, and transcript for each utterance. The CSV files are read in the training script and composed to a PyTorch-compatible dataset. The same processing pipeline can be used to prepare the data for all our recipes.
The data preparation steps consist of creating mappings used to convert acronyms in the Swbd corpus into acronyms based on the Fisher corpus convention, as well as cleaning the transcripts (e.g. removing markings and partial words).

2.3 CNN-RNN

The first ASR recipe is a combination of convolutional neural networks (CNNs), recurrent neural networks (RNNs), and linear layers applied in an encoder-decoder architecture. The model is trained on both CTC loss and Kullback-Leibler divergence between negative-log likelihood targets. Decoding is performed with a CTC beam search.
The encoder part of the system is comprised of two CNN blocks followed by a maximum pooling layer, a bidirectional LSTM [16], and two linear blocks (cf. left part of Figure 1). Each CNN block has two 2D convolutional layers and layer normalization [17], as well as maximum pooling and dropout [18]. Each linear block consists of a linear layer, batch normalization [19] and dropout.
The bidirectional LSTM has four layers with 1024 hidden units each, dropout with probability $p = 0.15$ for zeroing out elements of the input tensor.
The decoder consists of a Gated Recurrent Unit (GRU) cell [20], a location-aware attention block [21], and a linear layer to aggregate the outputs. The GRU cell, which is based on two multiplicative gates only, aims to simplify the more complex LSTM cell design. The location-aware attention block receives the outputs of the encoder, as well as the outputs of the GRU cell. The outputs of the GRU cell are concatenated with the outputs of the attention block and passed to a linear projection layer. The decoder outputs are processed by a final linear layer and a log softmax activation. The output after log softmax activation p_{CTC} is used to compute the CTC loss. The decoder output is passed through another linear layer followed by log softmax activation. The resulting log-probabilities p_{S2S} are used to compute the Kullback-Leibler loss.
By default, the model is trained for 20 epochs with a batch size of 8 and a CTC weight of 0.5. The Adadelta optimizer [22] is used with initial learning rate $\alpha = 1.0$ and decay rate $\rho = 0.95$. The learning rate is annealed based on the validation performance.

2.4 Transformer

The transformer recipe employs an attention-based encoder-decoder architecture [10]. The model is trained on both CTC loss and Kullback-Leibler divergence between negative-log likelihood targets. Decoding is performed with a joint CTC/attention beam search coupled with a transformer LM that is used on top of the decoder probabilities.
The composition of the transformer system is depicted in the upper right part of Figure 1. For each input sequence, a convolutional neural network (CNN) consisting of 3 layers is applied to reduce the length of the hidden representations before passing the input to the transformer model. The transformer architecture follows [3]. It employs 12 identical encoder blocks and 6 identical decoder blocks with 4 heads in the multi-head attention modules. Information about

the token position is encoded via sinusoidal positional encodings in 256-dimensional input embeddings. Each encoder and decoder block consists of two sub-layers: a multi-head attention mechanism and a position-wise fully connected feedforward network. Dropout and layer normalization [17] are applied after each sub-layer.
The transformer-encoder output is passed to the decoder and to a linear layer followed by log softmax activation. The output after log softmax activation p_{CTC} is used to compute the CTC loss. The transformer-decoder output is passed through another linear layer followed by log softmax activation. The resulting log-probabilities p_{S2S} are used to compute the Kullback-Leibler loss.
During decoding, the acoustic model is coupled with a transformer language model (LM). Its architecture is also based on [3]. The LM employs 12 encoder blocks but no decoder blocks. The LM encoder blocks use 264-dimensional input embeddings, a hidden layer size of 1024 and 12 attention heads.
By default, the LM is trained on the Swbd transcripts and on the transcripts provided by the Fisher corpus. The default LM weight is set to 0.3 and the beam size of the decoder is 60. The transformer AM is trained for 100 epochs with an effective batch size of 256 using gradient accumulation with a factor of 2 and a CTC weight of 0.3. The Adam optimizer [23] is used with exponential decay rates of $\beta_1 = 0.90$, $\beta_2 = 0.98$ and an initial learning rate of 6×10^{-3}. The learning rate is increased linearly for the first 25k steps and then decreased proportionally to the inverse square root of the step number [3].

2.5 Wav2vec 2.0

The third ASR recipe is the least complex one. It uses a pre-trained wav2vec 2.0 (W2V2) model for feature extraction, followed by linear encoder blocks for feature encoding. The W2V2 architecture used for acoustic feature extraction is described in [4]. It consists of a convolutional feature encoder, which is comprised of multiple identical blocks using temporal convolution, layer normalization, and a GELU activation function, followed by 12 transformer encoder blocks. The convolutional feature encoder generates a sequence of embeddings for each utterance, which are then passed to the transformer encoder to capture information about the entire input sequence. For self-supervised training, the output of the convolutional feature encoder is discretized to a finite set of speech representations using product quantization. Our recipe uses a large pre-trained model finetuned on 960 hours of Libri-Light [24] and LibriSpeech data.
The W2V2 feature extractor yields approximately 50 acoustic representations in $\mathbb{R}^{1024}$ for 1 second of raw audio input. The extracted acoustic features are passed through three encoder blocks consisting of a linear layer, batch normalization [19], leakyReLU activation and dropout [25] (cf. lower right part of Figure 1). A final linear layer yields outputs with the size of the vocabulary (1000 subword units). Greedy decoding on the logits and application of the CTC rules (removal of blank symbols and duplicates) yields the best path for each utterance.
By default, the model is trained with batch size 6 for 30 epochs. SpecAugment [26] is applied in the time-domain and the input audio is speed-perturbed at 95% and 105% of the regular utterance speed. The input audio data is also resampled to 16 kHz, to make it compatible with the pre-trained W2V2 feature extractor. The Adam [23] optimizer is used for the W2V2 part of the ystem with exponential decay rates of $\beta_1 = 0.90$, $\beta_2 = 0.99$ and an initial learning rate of 10^{-3}. The other parts of the model are trained using the Adadelta optimizer with an initial learning rate of $\alpha = 1.0$ and decay rate of $\rho = 0.95$. The learning rate is annealed based on the validation performance.

3 Experiments

We conducted ASR experiments using the architectures and hyperparameters described in section 2. The word error rates (WERs) on the Switchboard, Callhome and Eval2000 test sets are listed in Table 1. The recipes using W2V2 and transformer models achieved results comparable or better than those provided by Kaldi and ESPNet. The system based on W2V2 performed slightly better than ESPNet on the Callhome (14.67%) and Eval2000 (11.78%) corpora, but slightly worse on the Switchboard (8.76%) test set.
We combined the transformer-based system with two different transformer language models: the first LM was trained exclusively on the transcriptions of the Swbd and Fisher corpora, the second LM was trained on LibriSpeech data and was subsequently finetuned on the Swbd and Fisher corpora. The domain-specific LM trained only on the Swbd and Fisher data performed slightly better than the finetuned LM (13.99% vs. 14.34% on the Eval2000 test set).

Table 1 – Comparison of word error rates between our recipes and those from Kaldi and ESPNet.

Model	Switchboard	Callhome	Eval2000
wav2vec 2.0 (no LM)	8.76	14.67	11.78
Transformer (with Swbd+Fisher LM)	9.64	17.97	13.99
Transformer (with finetuned LM)	9.99	18.98	14.34
CNN-RNN (no LM)	15.01	24.44	19.90
Kaldi (`s5c`, "chain" AM, 4-gram LM)	9.80	19.40	14.70
ESPNet (`egs2/swbd/asr1`, conformer AM)	8.40	15.60	12.00

4 Usage

In addition to providing recipes for training the models from scratch, we also shared three pre-trained models coupled with the hosted inference API on HuggingFace. Testing these models is as easy as uploading or recording a speech sample on the HuggingFace website (see Figure 2).

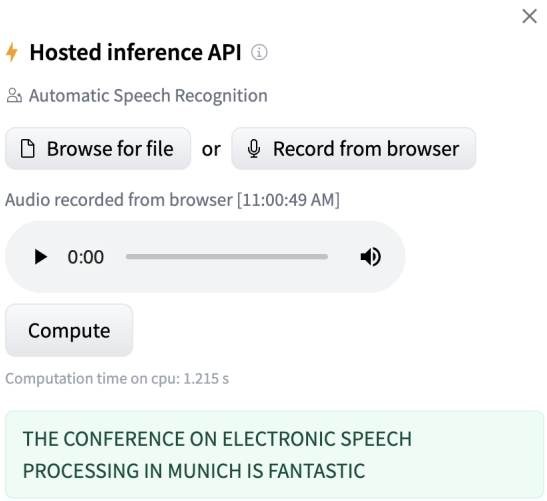

Figure 2 – Screenshot of the hosted inference application programming interface (API) on HuggingFace for the pre-trained CTC model after inference of a sample recorded by the browser. Available at `https://huggingface.co/speechbrain/asr-wav2vec2-switchboard`.

5 Conclusions

We contributed recipes for tokenization, language modeling and speech recognition experiments on the Switchboard corpus to the SpeechBrain project and shared three pre-trained models on the HuggingFace platform. ASR results show that our models yield better performance than those provided by Kaldi and are comparable to those of ESPNet. The main advantages of our recipes over those provided by other toolkits, are the complete removal of Kaldi-based preparation scripts, and the easy access via the hosted inference API. Future work will focus on further finetuning the models and the development of a recipe for German speech recognition.

References

[1] RAVANELLI, M., T. PARCOLLET, P. PLANTINGA, A. ROUHE ET AL.: *SpeechBrain: A general-purpose speech toolkit*. 2021. ArXiv:2106.04624, 2106.04624.

[2] PASZKE, A., S. GROSS, F. MASSA, A. LERER, J. BRADBURY, G. CHANAN, T. KILLEEN, Z. LIN, N. GIMELSHEIN, L. ANTIGA, A. DESMAISON, A. KOPF, E. YANG, Z. DEVITO, M. RAISON, A. TEJANI, S. CHILAMKURTHY, B. STEINER, L. FANG, J. BAI, and S. CHINTALA: *Pytorch: An imperative style, high-performance deep learning library*. In *Advances in Neural Information Processing Systems 32*, pp. 8024–8035. 2019.

[3] VASWANI, A., N. SHAZEER, N. PARMAR, J. USZKOREIT, L. JONES, A. N. GOMEZ, L. KAISER, and I. POLOSUKHIN: *Attention is all you need*. In *Advances in Neural Information Processing Systems*, vol. 30. 2017.

[4] BAEVSKI, A., Y. ZHOU, A. MOHAMED, and M. AULI: *wav2vec 2.0: A framework for self-supervised learning of speech representations*. In *Advances in Neural Information Processing Systems*, vol. 33, pp. 12449–12460. 2020.

[5] GODFREY, J. J. and HOLLIMAN, E.: *Switchboard-1 release 2*. 1993. doi:10.35111/SW3H-RW02. URL https://catalog.ldc.upenn.edu/LDC97S62.

[6] WATANABE, S., T. HORI, S. KARITA ET AL.: *Espnet: End-to-end speech processing toolkit*. 2018. URL https://arxiv.org/abs/1804.00015.

[7] POVEY, D., A. GHOSHAL, G. BOULIANNE, L. BURGET ET AL.: *The kaldi speech recognition toolkit*. In *2011 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*. 2011.

[8] KUDO, T. and J. RICHARDSON: *SentencePiece: A simple and language independent subword tokenizer and detokenizer for neural text processing*. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pp. 66–71. ACL, 2018.

[9] PANAYOTOV, V., G. CHEN, D. POVEY, and S. KHUDANPUR: *Librispeech: An asr corpus based on public domain audio books*. In *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 5206–5210. 2015.

[10] WATANABE, S., T. HORI, S. KIM, J. R. HERSHEY, and T. HAYASHI: *Hybrid ctc/attention architecture for end-to-end speech recognition. IEEE Journal of Selected Topics in Signal Processing*, 11(8), pp. 1240–1253, 2017.

[11] KUDO, T. and J. RICHARDSON: *SentencePiece: A simple and language independent subword tokenizer and detokenizer for neural text processing*. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pp. 66–71. 2018.

[12] LINGUISTIC DATA CONSORTIUM: *2000 hub5 english evaluation speech*. 2002. doi:10.35111/P7PZ-X179. URL https://catalog.ldc.upenn.edu/LDC2002S09.

[13] CANAVAN, A., D. GRAFF, and G. ZIPPERLEN: *Callhome american english speech*. 1997. doi:10.35111/EXQ3-X930. URL https://catalog.ldc.upenn.edu/LDC97S42.

[14] CIERI, C., D. GRAFF, O. KIMBALL, D. MILLER, and K. WALKER: *Fisher english training speech part 1 transcripts*. 2004. doi:10.35111/W4BK-9B14. URL https://catalog.ldc.upenn.edu/LDC2004T19.

[15] CIERI, C., D. GRAFF, O. KIMBALL, D. MILLER, and K. WALKER: *Fisher english training part 2, transcripts*. 2005. doi:10.35111/0YYQ-CJ29. URL https://catalog.ldc.upenn.edu/LDC2005T19.

[16] GRAVES, A. and J. SCHMIDHUBER: *Framewise phoneme classification with bidirectional lstm networks*. In *Proceedings. 2005 IEEE International Joint Conference on Neural Networks, 2005.*, vol. 4, pp. 2047–2052 vol. 4. 2005.

[17] BA, J. L., J. R. KIROS, and G. E. HINTON: *Layer normalization*. 2016. URL https://arxiv.org/abs/1607.06450.

[18] HINTON, G. E., N. SRIVASTAVA, A. KRIZHEVSKY, I. SUTSKEVER, and R. R. SALAKHUTDINOV: *Improving neural networks by preventing co-adaptation of feature detectors*. 2012. URL https://arxiv.org/abs/1207.0580.

[19] IOFFE, S. and C. SZEGEDY: *Batch normalization: Accelerating deep network training by reducing internal covariate shift*. In *Proceedings of the 32nd International Conference on Machine Learning*, vol. 37 of *Proceedings of Machine Learning Research*, pp. 448–456. 2015.

[20] CHO, K., B. VAN MERRIËNBOER, D. BAHDANAU, and Y. BENGIO: *On the properties of neural machine translation: Encoder–decoder approaches*. In *Proceedings of SSST-8, Eighth Workshop on Syntax, Semantics and Structure in Statistical Translation*, pp. 103–111. ACL, 2014.

[21] CHOROWSKI, J. K., D. BAHDANAU, D. SERDYUK, K. CHO, and Y. BENGIO: *Attention-based models for speech recognition*. In C. CORTES, N. LAWRENCE, D. LEE, M. SUGIYAMA, and R. GARNETT (eds.), *Advances in Neural Information Processing Systems*, vol. 28. 2015.

[22] ZEILER, M. D.: *Adadelta: An adaptive learning rate method*. 2012. URL https://arxiv.org/abs/1212.5701.

[23] KINGMA, D. P. and J. BA: *Adam: A method for stochastic optimization*. In *3rd International Conference on Learning Representations, ICLR 2015*. 2015.

[24] KAHN, J., M. RIVIÈRE, W. ZHENG, E. KHARITONOV, Q. XU, and P. E. A. MAZARÉ: *Libri-light: A benchmark for asr with limited or no supervision*. In *ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 7669–7673. 2020.

[25] SRIVASTAVA, N., G. HINTON, A. KRIZHEVSKY, I. SUTSKEVER, and R. SALAKHUTDINOV: *Dropout: A simple way to prevent neural networks from overfitting*. *Journal of Machine Learning Research*, 15(56), pp. 1929–1958, 2014.

[26] PARK, D. S., W. CHAN, Y. ZHANG ET AL.: *Specaugment: A simple data augmentation method for automatic speech recognition*. *Interspeech 2019*, 2019.