EFFECTIVENESS OF SELF-SUPERVISED PRE-TRAINING FOR ASR

Alexei Baevski, Abdelrahman Mohamed

Facebook AI Research
{abaevski, abdo}@fb.com

ABSTRACT

We compare self-supervised representation learning algorithms which either explicitly quantize the audio data or learn representations without quantization. We find the former to be more accurate since it builds a good vocabulary of the data through vq-wav2vec [1] self-supervision approach to enable learning of effective representations in subsequent BERT training. Different to previous work, we directly fine-tune the pre-trained BERT models on transcribed speech using a Connectionist Temporal Classification (CTC) loss instead of feeding the representations into a task-specific model. We also propose a BERT-style model learning directly from the continuous audio data and compare pre-training on raw audio to spectral features. Fine-tuning a BERT model on 10 hour of labeled Librispeech data with a vq-wav2vec vocabulary is almost as good as the best known reported system trained on 100 hours of labeled data on test-clean, while achieving a 25% WER reduction on test-other. When using only 10 minutes of labeled data, WER is 25.2 on test-other and 16.3 on test-clean. This demonstrates that self-supervision can enable speech recognition systems trained on a near-zero amount of transcribed data.

1 Introduction

Building Automatic Speech Recognition (ASR) systems, typically requires a large volume of training data to represent different factors contributing to the creation of speech signals, e.g. background noise, recording channel, speaker identity, accent, emotional state, topic under discussion, and the language used in communication. The practical need for building ASR systems for new conditions with limited resources spurred a lot of work focused on unsupervised speech recognition and representation learning [2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14], in addition to semi- and weakly-supervised learning techniques to reduce the supervised data needed in real-world scenarios [15, 16, 17, 18, 19, 20].

Recently impressive results have been reported for representation learning, that generalizes to different downstream tasks, through self-supervised learning for NLP and speech [21, 22, 11, 14, 1]. Self-supervised representation learning tasks include predicting masked parts of the input, reconstructing inputs through low bit-rate channels, or contrasting similar data points against different ones.

In this work we compare different approaches of self-supervised pre-training for speech data. We consider learning discrete units to represent the audio data through either self-supervision [1] or through clustering spectral features, followed by pre-training over these units using a bi-directional transformer (BERT) model [21]. This is compared to directly learning representations without explicit quantization over the raw audio as well as spectral features. Previous work fed the representations of the pre-trained model into a task-specific architecture for speech recognition instead of the raw waveform [14, 1]. Instead, similar to BERT, we achieved lower WER by directly fine-tuning the pre-trained model on transcribed speech data using a CTC loss. Our experiments demonstrate that discrete unit discovery, followed by BERT training achieves better results than representations learned without explicit quantization. Disentangling acoustic unit discovery from learning the sequential relationship between them, enables better representations of the data which in turn improves down-stream model accuracy.

We pre-train our models on the unlabeled 960h Librispeech [23] data and follow the Libri-light [24] limited resource supervised training sets of 10 hours, 1 hour, and 10 mins. Our best model fine-tuned only on 1 hour of labeled data can outperform the best known result from the literature relying on 100h of labeled data [25] on the standard Librispeech test-other subset. Using only 10 minutes of labeled data the approach achieves 16.3/25.2 WER on test-clean/other.

2 Approach

2.1 Discrete BERT

Our work builds on the recently proposed vq-wav2vec approach [1] where audio is quantized using a contrastive loss, then features learned on top by a BERT model [21]. For the vq-wav2vec quantization, we use the gumbel-softmax variant with the same setup as described in [1]. This model quantizes the Librispeech dataset into 13.5k unique codes.

To understand the impact of discrete acoustic representations of vq-wav2vec [1], as alternatives, we explore quantizing the standard mel-frequency cepstral coefficients (MFCC) and log-mel filterbanks coefficients (FBANK), choosing a subset small enough to fit into GPU memory and running k-means with 13.5k centroids (to match the vq-wav2vec setup) to convergence. We then assign the index of the closest centroid to represent each time-step.

We train a standard BERT model [21, 26] with only the

masked language modeling task on each set of inputs in a similar way as described in [1], namely by choosing tokens for masking with probability of 0.05, expanding each chosen token to a span of a length sampled from a normal distribution with mean 10 and standard deviation 10 (spans may overlap) and then computing a cross-entropy loss which attempts to maximize the likelihood of predicting the true token for each one that was masked.

Following [27], we replace the fixed positional embeddings in the BERT model with a single group convolutional layer that is applied directly on the embeddings before any of the transformer blocks. The convolutional layer has a kernel size of 128 and group size of 16 to reduce the number of added parameters.

2.2 Continuous BERT

A masked language modeling task cannot be performed with continuous inputs and outputs, as there are no targets to predict in place of the masked tokens. Instead of reconstructing the input as in [28], we classify the masked positive example among a set of negatives. The inputs to the model are dense wav2vec features [14], MFCC or FBANK features representing 10ms of audio data. Some of these inputs are replaced with a *mask* embedding and are then fed into a transformer encoder. We then compute the dot product between the outputs corresponding to each masked input, the true input that was masked, and a set of negatives sampled from other masked inputs within the same batch. The model is optimized with the InfoNCE loss [11] where given one positive sample $\mathbf{z}_i$ and N negative samples $\tilde{\mathbf{z}}$ we minimize:

$$\mathcal{L}_k = \sum_{i=1}^{T} \frac{\exp(\mathbf{z}_i)}{\sum_{j=1}^{N} \exp(\tilde{\mathbf{z}}^j)} \tag{1}$$

where each sample $\mathbf{z}_i$ is computed as a dot product of the output of the model at timestep i and the true unmasked value of positive example at timestep i or a randomly sampled negative example. To stabilize training, we add the squared sum of logits produced by the dot-product to the loss, and then apply a soft clamp $\hat{s_i} = \lambda \tanh(s_i/\lambda)$ for each logit s_i to prevent the model's tendency to continually increase the magnitude of logits during training [29]. We use the same kind of convolutional positional layer as described in section 2.1.

2.3 Supervised fine-tuning

The pre-trained models are fine-tuned to perform the ASR task by adding a randomly initialized linear projection on top of the features computed by the transformer models into V classes representing the vocabulary of the task. The vocabulary is 29 tokens for character targets plus a word boundary token. The models are optimized by minimizing the CTC loss. We apply SpecAugment [30] inspired masking to time-steps and channels during training which delays overfitting and significantly improves the final accuracy numbers, especially on the smallest subsets.

We train a single seed for all subsets except the 10 minute one, where we train 5 seeds and choose the best one. For 10 minute subset, some of the seeds fail by entering the overfit regime very early. We train on a single GPU using the Adam optimizer with a tri-state learning scheduler where the learning rate is linearly increased from 1e-7 to 2e-05 in the first stage, held at 2e-5 in the second, and finally linearly decayed to 0 in the third. For 1h and 10min subsets we train for 1250 / 6600 / 12150 updates in each respective stage, for 10h subset we train for 5000 / 16500 / 28500 updates and for 100h we train for 8000 / 52800 / 91200 updates. We use a batch size of 6144 timesteps (61.44 seconds worth of audio) for the 100 hour subset and 3072 timesteps for other subsets.

During fine-tuning, we apply a modified SpecAugment [30] policy, where we randomly choose a number of starting timesteps to mask, with each timestep having a chance of 3.75% of being chosen. A span of 20 timesteps starting at each of the chosen position is then replaced with the mask embedding used during unsupervised training (spans may overlap). We also apply channel masking, in which we choose the starting channel index from all channels with a probability of 0.4% and the length of the channel mask by sampling from a normal distribution with a mean of 64 and standard deviation of 64. The chosen (and possibly overlapping) spans of channels are then zeroed out.

We apply a dropout at every layer of the transformer of 0.1 for 10 minute and 1 hour setup, but we disable it for other subsets as masking described above appears to provide enough regularization.

3 Experiments

We implement our models in the fairseq [31] toolkit.

3.1 Data

All experiments are performed by pre-training on the 960 hours of audio only data of the Librispeech [23] training set, fine-tuning on the Libri-light [24] limited resource supervised training sets of 10 hours (24 speakers), 1 hour (24 speakers), and 10 minutes (4 speakers). The Libri-light training sets are sampled equally from the two clean and noisy portions, a balance of male and female speakers. We also report results of models fine-tuned on 100 hours following the "train-clean-100" subset. All models are evaluated on the standard Librispeech dev and test splits.

3.2 Models

3.2.1 Quantized Inputs Training

We first train the vq-wav2vec quantization model following the gumbel-softmax setup described in [1]. After training this model on 960h of Librispeech and quantizing the training dataset, we are left with 13.5k unique codewords combinations.

For quantizing MFCC and FBANK features extracted using the Kaldi [32] toolkit, we use 8 Volta GPUs with 32GB memory to compute 13.5k K-Means centroids matching the number of unique tokens produced by the vq-wav2vec model. To fit into GPU memory, we subsample 50% of MFCC features and 25% of FBANK features from the training set before

running the K-Means algorithm.

The model we use for the masked language modeling task is a standard BERT model with 12 transformer layers, model dimension 768, inner dimension (FFN) 3072 and 12 attention heads [21]. The learning rate is warmed up over the first 10,000 updates to a peak value of 1×10^{-5}, and then linearly decayed over a total of 250k updates. We train on 128 GPUs with a batch size of 3072 tokens per GPU giving a total batch size of 393k tokens [33] where each token represents 10ms of audio data.

To mask the input sequence, we follow [1] and randomly sample $p = 0.05$ of all tokens to be a starting index, without replacement, and mask M consecutive tokens from every sampled index; spans may overlap. M is sampled from a Gaussian distribution with $\mu = 10$ and $\sigma = 10$, rounded to the nearest integer greater than or equal to zero.

Different from [1], we do not concatenate different utterances to form examples for training, instead each utterance is treated as a single example, as we find that this approach produces better results after fine-tuning.

3.2.2 Continuous Inputs Training

For training on dense features, we use a model similar to a standard BERT model with the same parameterization as the one used for quantized input training, but we use the wav2vec, MFCC or FBANK inputs directly. We add 128 relative positional embeddings at every multi-head attention block [34] instead of fixed positional embeddings to make it easier to handle longer examples. We train this model on 8 GPUs with a batch size of 9600 inputs per GPU, resulting in a total batch size of 76,800. We find that increasing the number of GPUs (which increases the effective batch size) does not lead to better results with this particular setup.

Wav2vec features are 512-dimensional, while MFCC features have 39 dimensions and FBANK features have 80. We introduce a simple linear projection from the feature dimension to BERT dimension (768) for all models.

Similarly to the approach in 3.2.1, we choose time-steps to mask by randomly sampling, without replacement, $p = 0.05$ of all time-steps to be a starting index, and mask M consecutive time-steps from every sampled index; spans may overlap, where M is sampled from a Gaussian distribution with $\mu = 10$ and $\sigma = 10$, rounded to the nearest integer greater than or equal to zero. We sample 10 negative examples from other masked time-steps from the same example, and an additional 10 negative examples from masked time-steps occurring anywhere in the batch. We compute a dot product between the original features and the output corresponding to the same time-step after they are processed by the BERT model. We add the squared sum of logits from these computations multiplied by $\lambda = 0.04$ to the loss, and then apply a smooth clamp by recomputing each logit $\hat{s_i} = 20 \tanh(s_i/20)$.

The learning rate is warmed up over the first 10,000 updates to a peak value of 1×10^{-5}, and then linearly decayed over a total of 250k updates.

3.3 Methodology

For quantized inputs, we compute token indices using the gumbel-softmax based vq-wav2vec model. For MFCC and FBANK features we take the index of the closest centroid, as measured by finding the minimum Euclidean distance, to each corresponding feature in the Librispeech dataset. We then train a BERT model as descirbed in §3.2.1.

For wav2vec continuous inputs, we use features extracted by the publicly available wav2vec [14] model which contains 6 convolutional blocks in the feature extractor and 11 convolutional blocks in the aggregator module. We use the outputs of the aggregator as features. For MFCC and FBANK, we use those features directly after applying a single linear projection to upsample them to the model dimensionality.

We fine-tune our pre-trained models on either 100 hours of Librispeech train-clean-100 subset, 10 hours, 1 hour, or 10 minutes of labelled data following Libri-light [24] training sets. We use the standard CTC loss and train for up to 20k updates. We find that the pre-trained models converge after only around 4k updates, while the models trained from scratch tend to converge much later, around 18k updates. We fine-tune all models with a learning rate of 0.0001 that is linearly warmed up over the first 2k updates and then annealed following a cosine learning rate schedule over the last 18k updates. We set the dropout of the pre-trained BERT models to 0.1 and sweep on dropout of the BERT model outputs before the final projection layer over values between 0.0 and 0.4 in increments of 0.1. For each model, we choose a single best checkpoint that has the best loss on the validation set, which is a combination of dev-clean and dev-other standard Librispeech splits.

We use the publicly available wav2letter++ [35] decoder integrated into the Fairseq framework with the official Librispeech 4-gram language model. We run a sweep on weights for language model score, word score and silence token weights for each model, where parameters are chosen randomly and evaluated on the dev-other Librispeech set. We use the weights found by these sweeps to evaluate and report results for all other splits. The sweeps are run with beam size of 250, while the final decoding uses a beam size of 1500.

3.4 Results

In our first experiment, we compare unit discovery followed by BERT training over the resulting discrete units (Discrete BERT) to directly learning representations from the audio inputs (Continuous BERT) in different simulated labeled data scenarios ranging from 100 hours to 10 minutes. We compare quantization with vq-wav2vec to clustered MFCC and FBANK features. The continuous BERT variant learns directly from the audio representations without explicit quantization and we experiment with inputting wav2vec, MFCC and FBANK features.

Table 1 compares WERs of different input features and pre-training methods on the standard Librispeech clean and other subsets. The first observation is that Discrete BERT outperforms Continuous BERT in all settings. This shows that pre-training over meaningful discrete units outperforms di-

Model	Input features	dev clean	dev other	test clean	test other
10 mins of labeled data					
Discrete BERT	vq-wav2vec	**15.7**	**24.1**	**16.3**	**25.2**
	MFCC	30.3	48.8	31.5	49.1
	FBANK	35.7	53.9	35.5	55.0
Continuous BERT	wav2vec	49.1	66.0	49.5	66.3
	MFCC	82.2	91.7	80.4	90.4
	FBANK	92.9	96.1	92.3	95.9
1 hour of labeled data					
Discrete BERT	vq-wav2vec	**8.5**	**16.4**	**9.0**	**17.6**
	MFCC	14.1	32.1	14.3	32.5
	FBANK	14.2	30.6	14.6	31.7
Continuous BERT	wav2vec	22.1	42.0	22.4	44.0
	MFCC	53.8	75.0	52.8	74.5
	FBANK	56.7	76.2	55.0	76.1
10 hours of labeled data					
Discrete BERT	vq-wav2vec	**5.3**	**13.2**	**5.9**	**14.1**
	MFCC	9.8	26.6	9.9	27.8
	FBANK	9.8	25.7	10.1	26.6
Continuous BERT	wav2vec	13.6	31.7	14.1	34.3
	MFCC	27.5	54.2	27.4	55.7
	FBANK	25.0	50.2	24.9	51.7
100 hours of labeled data					
Discrete BERT	vq-wav2vec	**4.0**	**10.9**	**4.5**	**12.1**
	MFCC	7.6	24.2	7.8	24.4
	FBANK	7.2	22.8	7.8	23.3
Continuous BERT	wav2vec	11.3	26.4	11.8	28.3
	MFCC	11.6	34.0	12.4	35.5
	FBANK	10.1	30.9	11.0	31.8

Table 1: WERs of Discrete and Continuous BERT models using different input representations and labeled data sizes.

rectly learning representations from the continuous unlabeled data. The initial unit discovery builds a vocabulary that makes the subsequent BERT pre-training more effective.

The best input features are obtained through self-supervised learning through vq-wav2vec [1] for Discrete BERT, or wav2vec [14] for Continuous BERT.

For Discrete BERT, vq-wav2vec provides about 40% of relative error reduction for both test sutsets compared to clustered spectral features across all training set sizes, with bigger gains on the noisy test-other subset.

Pre-training brings clear benefits: When reducing the amount of labeled training data from 100h to 10h results in an increase of only 2 WER on test-other and 1.4 WER on test-clean for Discrete BERT with vq-wav2vec inputs. This shows that pre-training is effective and particularly so when little labeled data is available. When reducing the amount of labeled data to only 10 minutes, Discrete BERT with va-wav2vec inputs can still achieve a WER of 16.3/25.2 on test-clean/other.

	Labeled data	test clean	test other
Wang et al. (2019) [36]	960h	2.6	5.6
Irie et al. (2019) [37] (No LM)	100h	12.9	35.5
Kahn et al. (2019) [38] (Conv LM)	100h	5.9	24.1
Panayotov et al. (2015) [23]	100h	6.6	22.5
Lüscher et al. (2019) [25]	100h	5.8	18.6
Kawakami et al. (2019) [39]	96h	9.4	26.8
vq-wav2vec + Discrete BERT (Ours)	10m	16.3	25.2
	1h	9.0	17.6
	10h	5.9	14.1
	100h	4.5	12.1

Table 2: Comparison to previously published results. All results use 4-gram language models, except [37, 38].

Table 2 shows a comparison of Discrete BERT to results from the literature. Fine-tuning Discrete BERT on only 10 hour of labeled data can match the best known result on 100 hours of labeled Librispeech data [25] on test-clean, while achieving a 25% relative WER reduction on test-other. Moreover, when using the same train-clean-100 subset for fine-tuning, Discrete BERT with vq-wav2vec inputs improves by 6.5 WER (35% relative WER reduction) on test-other and 1.3 WER (22% relative WER reduction) on test-clean over [25].

The closest setup to ours is [39] which learns representations using CPC [11] and then feed these into an ASR system trained on about 96h of labeled data achieving 9.4/26.8% on test-clean/other, which is surpassed by our Discrete-BERT model trained on vq-wav2vec representations fine-tuned on 10mins and 1 hour.

4 Conclusion and Future work

We presented a systematic comparison of self-supervised pre-training approaches for speech recognition. The most effective method is to first learn a discrete vocabulary of the data with vq-wav2vec followed by standard BERT training over these discrete units. This performs much better than directly learning from the continuous audio data. Different to previous work which relied on task-specific ASR models, we directly fine-tune the resulting BERT model on transcribed speech data to act as speech recognition models. This approach can achieve better accuracy on test-other than the best known result with 100 hours of labeled data while relying on two orders magnitude less labeled data. When the model is fine-tuned on only 10 minutes of data, it can still achieve a WER 25.2 on test-other and WER 16.3 on test-clean.

5 References

[1] Alexei Baevski, Steffen Schneider, and Michael Auli, "vq-wav2vec: Self-supervised learning of discrete speech representations," in *Proc. of ICLR*, 2020.

[2] A. Park and J. Glass, "Unsupervised pattern discovery in speech," *IEEE TASLP*, vol. 16, no. 1, pp. 186–197, 2008.

[3] Aren Jansen, Kenneth Church, and Hynek Hermansky, "Towards spoken term discovery at scale with zero resources.," in *Proc. of Interspeech*, 2010.

[4] J. Glass, "Towards unsupervised speech processing," in *ISSPA*, 2012.

[5] Aren Jansen, Emmanuel Dupoux, Sharon Goldwater, Mark Johnson, Sanjeev Khudanpur, Kenneth Church, Naomi Feldman, Hynek Hermansky, Florian Metze, Richard Rose, et al., "A summary of the 2012 JHU CLSP workshop on zero resource speech technologies and models of early language acquisition," in *Proc. of ICASSP*, 2013.

[6] Keith Levin, Katharine Henry, Aren Jansen, and K. Livescu, "Fixed-dimensional acoustic embeddings of variable-length segments in low-resource settings," in *Proc. of ASRU*, 2013.

[7] Samy Bengio and Georg Heigold, "Word embeddings for speech recognition," in *Proc. of Interspeech*, 2014.

[8] Yu-An Chung, Chao-Chung Wu, Chia-Hao Shen, Hung yi Lee, and L. Lee, "Audio word2vec: Unsupervised learning of audio segment representations using sequence-to-sequence autoencoder," in *Proc. of Interspeech*, 2016.

[9] Odette Scharenborg, Laurent Besacier, Alan Black, Mark Hasegawa-Johnson, Florian Metze, Graham Neubig, Sebastian Stüker, Pierre Godard, Markus Müller, Lucas Ondel, et al., "Linguistic unit discovery from multi-modal inputs in unwritten languages: Summary of the "speaking rosetta" JSALT 2017 workshop," in *Proc. of ICASSP*, 2018.

[10] Yu-An Chung and James Glass, "Speech2vec: A sequence-to-sequence framework for learning word embeddings from speech," in *Proc. of Interspeech*, 2018.

[11] Aäron van den Oord, Yazhe Li, and Oriol Vinyals, "Representation learning with contrastive predictive coding," *arXiv*, 2018.

[12] Yu-An Chung, Wei-Ning Hsu, Hao Tang, and James R. Glass, "An unsupervised autoregressive model for speech representation learning," in *Proc. of Interspeech*, 2019.

[13] Shane Settle, Kartik Audhkhasi, Karen Livescu, and Michael Picheny, "Acoustically grounded word embeddings for improved acoustics-to-word speech recognition," in *Proc. of ICASSP*, 2019.

[14] Steffen Schneider, Alexei Baevski, Ronan Collobert, and Michael Auli, "wav2vec: Unsupervised pre-training for speech recognition," in *Proc. of Interspeech*, 2020.

[15] K. Vesely, M. Hannemann, and L. Burget, "Semi-supervised training of deep neural networks," in *Proc. of ASRU*, 2013.

[16] Sheng Li, Xugang Lu, Shinsuke Sakai, Masato Mimura, and Tatsuya Kawahara, "Semi-supervised ensemble dnn acoustic model training," in *Proc. of ICASSP*, 2017.

[17] S. Krishnan Parthasarathi and N. Strom, "Lessons from building acoustic models with a million hours of speech," in *Proc. of ICASSP*, 2019.

[18] Bo Li, Ruoming Pang, Tara Sainath, and Zelin Wu, "Semi-supervised training for end-to-end models via weak distillation," in *Proc. of ICASSP*, 2019.

[19] Grzegorz Chrupała, Lieke Gelderloos, and Afra Alishahi, "Representations of language in a model of visually grounded speech signal," in *Proc. of ACL*, 2017.

[20] Herman Kamper, Shane Settle, Gregory Shakhnarovich, and Karen Livescu, "Visually grounded learning of keyword prediction from untranscribed speech," in *Proc. of Interspeech*, 2017.

[21] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," in *Proc. of NAACL*, 2019.

[22] Alexei Baevski, Sergey Edunov, Yinhan Liu, Luke Zettlemoyer, and Michael Auli, "Cloze-driven pretraining of self-attention networks," in *Proc. of EMNLP*, 2019.

[23] Vassil Panayotov, Guoguo Chen, Daniel Povey, and Sanjeev Khudanpur, "Librispeech: An asr corpus based on public domain audio books," in *Proc. of ICASSP*, 2015.

[24] Jacob Kahn et al., "Libri-light: A benchmark for asr with limited or no supervision," *arXiv preprint arXiv:1912.07875*, 2019.

[25] Christoph Lüscher, Eugen Beck, Kazuki Irie, Markus Kitza, Wilfried Michel, Albert Zeyer, Ralf Schlüter, and Hermann Ney, "Rwth asr systems for librispeech: Hybrid vs attention," in *Interspeech 2019*, 2019.

[26] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov, "Roberta: A robustly optimized bert pretraining approach," *arXiv*, vol. abs/1907.11692, 2019.

[27] Abdelrahman Mohamed, Dmytro Okhonko, and Luke Zettlemoyer, "Transformers with convolutional context for asr," *arXiv*, 2019.

[28] Aaron van den Oord, Oriol Vinyals, and Koray Kavukcuoglu, "Neural discrete representation learning," *arXiv*, vol. abs/1711.00937, 2017.

[29] Philip Bachman, R Devon Hjelm, and William Buchwalter, "Learning representations by maximizing mutual information across views," *arXiv*, vol. abs/1906.00910, 2019.

[30] Daniel S. Park, William Chan, Yu Zhang, Chung-Cheng Chiu, Barret Zoph, Ekin D. Cubuk, and Quoc V. Le, "Specaugment: A simple data augmentation method for automatic speech recognition," in *Proc. of Interspeech*. 2019, ISCA.

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

[32] Daniel Povey, Arnab Ghoshal, Gilles Boulianne, Lukas Burget, Ondrej Glembek, Nagendra Goel, Mirko Hannemann, Petr Motlicek, Yanmin Qian, Petr Schwarz, Jan Silovsky, Georg Stemmer, and Karel Vesely, "The kaldi speech recognition toolkit," in *Proc. of ASRU*, 2011.

[33] Myle Ott, Sergey Edunov, David Grangier, and Michael Auli, "Scaling neural machine translation," in *Proc. of WMT*, 2018.

[34] Zihang Dai, Zhilin Yang, Yiming Yang, Jaime Carbonell, Quoc V. Le, and Ruslan Salakhutdinov, "Transformer-xl: Attentive language models beyond a fixed-length context," *arXiv*, vol. abs/1901.02860, 2019.

[35] V. Pratap, A. Hannun, Q. Xu, J. Cai, J. Kahn, G. Synnaeve, V. Liptchinsky, and R. Collobert, "Wav2letter++: A fast open-source speech recognition system," in *Proc. of ICASSP*, 2019.

[36] Yongqiang Wang, Abdelrahman Mohamed, Duc Le, Chunxi Liu, Alex Xiao, Jay Mahadeokar, Hongzhao Huang, Andros Tjandra, Xiaohui Zhang, Frank Zhang, Christian Fuegen, Geoffrey Zweig, and Michael L. Seltzer, "Transformer-based acoustic modeling for hybrid speech recognition," *arXiv*, vol. abs/1910.09799, 2019.

[37] Kazuki Irie, Rohit Prabhavalkar, Anjuli Kannan, Antoine Bruguier, David Rybach, and Patrick Nguyen, "On the choice of modeling unit for sequence-to-sequence speech recognition," in *Interspeech 2019*, 2019.

[38] Jacob Kahn, Ann Lee, and Awni Hannun, "Self-training for end-to-end speech recognition," *arXiv*, vol. abs/1909.09116, 2019.

[39] Kazuya Kawakami, Luyu Wang, Chris Dyer, Phil Blunsom, and Aaron van den Oord, "Unsupervised learning of efficient and robust speech representations," 2019.