INTERSPEECH 2019
September 15–19, 2019, Graz, Austria

wav2vec: Unsupervised Pre-training for Speech Recognition

Steffen Schneider, Alexei Baevski, Ronan Collobert, Michael Auli

Facebook AI Research

Abstract

We explore unsupervised pre-training for speech recognition by learning representations of raw audio. wav2vec is trained on large amounts of unlabeled audio data and the resulting representations are then used to improve acoustic model training. We pre-train a simple multi-layer convolutional neural network optimized via a noise contrastive binary classification task. Our experiments on WSJ reduce WER of a strong character-based log-mel filterbank baseline by up to 36 % when only a few hours of transcribed data is available. Our approach achieves 2.43 % WER on the nov92 test set. This outperforms Deep Speech 2, the best reported character-based system in the literature while using three orders of magnitude less labeled training data.[1]

1. Introduction

Current state of the art models for speech recognition require large amounts of transcribed audio data to attain good performance [1]. Recently, pre-training of neural networks has emerged as an effective technique for settings where labeled data is scarce. The key idea is to learn general representations in a setup where substantial amounts of labeled or unlabeled data is available and to leverage the learned representations to improve performance on a downstream task for which the amount of data is limited. This is particularly interesting for tasks where substantial effort is required to obtain labeled data, such as speech recognition.

In computer vision, representations for ImageNet [2] and COCO [3] have proven to be useful to initialize models for tasks such as image captioning [4] or pose estimation [5]. Unsupervised pre-training for computer vision has also shown promise [6, 7]. In natural language processing (NLP), unsupervised pre-training of language models [8, 9, 10] improved many tasks such as text classification, phrase structure parsing and machine translation [11, 12]. In speech processing, pre-training has focused on emotion recogniton [13], speaker identification [14], phoneme discrimination [15, 16] as well as transferring ASR representations from one language to another [17]. There has been work on unsupervised learning for speech but the resulting representations have not been applied to improve supervised speech recognition [18, 19, 20, 21, 22].

In this paper, we apply unsupervised pre-training to improve supervised speech recognition. This enables exploiting unlabeled audio data which is much easier to collect than labeled data. Our model, wav2vec, is a convolutional neural network that takes raw audio as input and computes a general representation that can be input to a speech recognition system. The objective is a contrastive loss that requires distinguishing a true future audio sample from negatives [23, 24, 16]. Different to previous work [16], we move beyond frame-wise phoneme classification and apply the learned representations to improve strong supervised ASR systems. wav2vec relies on a fully convolutional architecture which can be easily parallelized over time on modern hardware compared to recurrent models used in previous work (§2).

Experimental results on the WSJ benchmark demonstrate that pre-trained representations estimated on about 1,000 hours of unlabeled speech can substantially improve a character-based ASR system and outperform the best character-based result in the literature, Deep Speech 2, improving WER from 3.1 % to 2.43 %. On TIMIT, pre-training enables us to match the best reported result in the literature. In a simulated low-resource setup with only eight hours of transcribed audio data, wav2vec reduces WER by up to 36 % compared to a baseline model that relies on labeled data only (§3, §4).

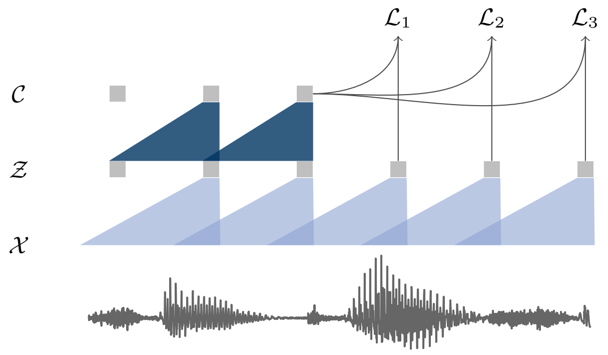

Figure 1: *Illustration of pre-training from audio data $\mathcal{X}$ which is encoded with two convolutional neural networks that are stacked on top of each other. The model is optimized to solve a next time step prediction task.*

2. Pre-training Approach

Given an audio signal as input, we optimize our model (§2.1) to predict future samples from a given signal context. A common problem with these approaches is the requirement to accurately model the data distribution $p(\mathbf{x})$, which is challenging. We avoid this problem by first encoding raw speech samples $\mathbf{x}$ into a feature representation $\mathbf{z}$ at a lower temporal frequency and then implicitly model a density ratio $p(\mathbf{z}_{i+k}|\mathbf{z}_i \dots \mathbf{z}_{i-r})/p(\mathbf{z}_{i+k})$ similar to [16].

2.1. Model

Our model takes raw audio signal as input and then applies two networks. The encoder network embeds the audio signal in a latent space and the context network combines multiple time-steps of the encoder to obtain contextualized representations (Figure 1). Both networks are then used to compute the objective function (§2.2).

[1]The code is available as part of fairseq (`https://github.com/pytorch/fairseq`).

Given raw audio samples $\mathbf{x}_i \in \mathcal{X}$, we apply the *encoder* network $f : \mathcal{X} \mapsto \mathcal{Z}$ parameterized as a five-layer convolutional network [16]. Alternatively, one could use other architectures such as the trainable frontend of [25]. The encoder layers have kernel sizes $(10, 8, 4, 4, 4)$ and strides $(5, 4, 2, 2, 2)$. The output of the encoder is a low frequency feature representation $\mathbf{z}_i \in \mathcal{Z}$ which encodes about 30 ms of 16 kHz of audio and the striding results in representations $\mathbf{z}_i$ every 10ms.

Next, we apply the *context* network $g : \mathcal{Z} \mapsto \mathcal{C}$ to the output of the encoder network to mix multiple latent representations $\mathbf{z}_i \dots \mathbf{z}_{i-v}$ into a single contextualized tensor $\mathbf{c}_i = g(\mathbf{z}_i \dots \mathbf{z}_{i-v})$ for a receptive field size v. The context network has nine layers with kernel size three and stride one. The total receptive field of the context network is about 210 ms.

The layers in both the encoder and context networks consist of a causal convolution with 512 channels, a group normalization layer and a ReLU nonlinearity. We normalize both across the feature and temporal dimension for each sample which is equivalent to group normalization with a single normalization group [26]. We found it important to choose a normalization scheme that is invariant to the scaling and the offset of the input. This choice resulted in representations that generalize well across datasets.

For training on larger datasets, we also consider a model variant ("wav2vec large") with increased capacity, using two additional linear transformations in the encoder and a considerably larger context network comprised of twelve layers with increasing kernel sizes $(2, 3, \dots, 13)$. We found it important to introduce skip connections in the aggregator to help convergence in this case. The total receptive field in the last context network layer is hereby increased to about 810 ms.

2.2. Objective

We train the model to distinguish a sample $\mathbf{z}_{i+k}$ that is k steps in the future from distractor samples $\tilde{\mathbf{z}}$ drawn from a proposal distribution p_n, by minimizing the contrastive loss for each step $k = 1, \dots, K$:

$$\mathcal{L}_k = -\sum_i \Big(\log \sigma(\mathbf{z}_{i+k}^\top h_k(\mathbf{c}_i)) + \lambda \mathop{\mathbb{E}}_{\tilde{\mathbf{z}} \sim p_n} [\log \sigma(-\tilde{\mathbf{z}}^\top h_k(\mathbf{c}_i))] \Big) \quad (1)$$

where we denote the sigmoid $\sigma(x) = 1/(1 + \exp(-x))$, and where $\sigma(\mathbf{z}_{i+k}^\top h_k(\mathbf{c}_i))$ is the probability of $\mathbf{z}_{i+k}$ being the true sample. We consider a step-specific affine transformation $h_k(\mathbf{c}_i) = W_k \mathbf{c}_i + \mathbf{b}_k$ for each step k, that is applied to $\mathbf{c}_i$ [16]. We optimize the loss $\mathcal{L} = \sum_{k=1}^{K} \mathcal{L}_k$, summing (1) over different step sizes. In practice, we approximate the expectation by sampling ten negatives examples by uniformly choosing distractors from each audio sequence, i.e., $p_n(\mathbf{z}) = \frac{1}{T}$, where T is the sequence length and we set λ to the number of negatives.[2]

After training, we input the representations produced by the context network $\mathbf{c}_i$ to the acoustic model instead of log-mel filterbank features.

3. Experimental Setup

3.1. Data

We consider the following corpora: For phoneme recognition on TIMIT [27] we use the standard train, dev and test split where the training data contains just over three hours of audio data. Wall Street Journal (WSJ; [28, 29]) comprises about 81 hours of transcribed audio data. We train on si284, validate on nov93dev and test on nov92. Librispeech [30] contains a total of 960 hours of clean and noisy speech for training. For pre-training, we use either the full 81 hours of the WSJ corpus, an 80 hour subset of clean Librispeech, the full 960 hour Librispeech training set or a combination of all of them.

To train the baseline acoustic model we compute 80 log-mel filterbank coefficients for a 25 ms sliding window with stride 10 ms. Final models are evaluated in terms of both word error rate (WER) and letter error rate (LER).

3.2. Acoustic Models

We use the wav2letter++ toolkit for training and evaluation of acoustic models [31]. For the TIMIT task, we follow the character-based wav2letter++ setup of [25] which uses seven consecutive blocks of convolutions (kernel size 5 with 1,000 channels), followed by a PReLU nonlinearity and a dropout rate of 0.7. The final representation is projected to a 39-dimensional phoneme probability. The model is trained using the Auto Segmentation Criterion (ASG; Collobert et al., 2016)) using SGD with momentum.

Our baseline for the WSJ benchmark is the wav2letter++ setup described in [33] which is a 17 layer model with gated convolutions [34]. The model predicts probabilities for 31 graphemes, including the standard English alphabet, the apostrophe and period, two repetition characters (e.g. the word `ann` is transcribed as `an1`), and a silence token (`|`) used as word boundary.

All acoustic models are trained on 8 NVIDIA V100 GPUs using the distributed training implementations of fairseq and wav2letter++. When training acoustic models on WSJ, we use plain SGD with learning rate 5.6 as well as gradient clipping [33] and train for 1,000 epochs with a total batch size of 64 audio sequences. We use early stopping and choose models based on validation WER after evaluating checkpoints with a 4-gram language model. For TIMIT we use learning rate 0.12, momentum 0.9 and we train for 1,000 epochs on 8 GPUs with a batch size of 16 audio sequences.

3.3. Decoding

For decoding the emissions from the acoustic model we use a lexicon as well as a separate language model trained on the WSJ language modeling data only. We consider a 4-gram KenLM language model [37], a word-based convolutional language model [33], and a character based convolutional language model [39]. We decode the word sequence $\mathbf{y}$ from the output of the context network $\mathbf{c}$ or log-mel filterbanks using the beam search decoder of [33] by maximizing

$$\max_{\mathbf{y}} f_{\text{AM}}(\mathbf{y}|\mathbf{c}) + \alpha \log p_{\text{LM}}(\mathbf{y}) + \beta |\mathbf{y}| - \gamma \sum_{i=1}^{T} [\pi_i = \text{`}|\text{'}] \quad (2)$$

where f_{AM} is the acoustic model, p_{LM} is the language model, $\pi = \pi_1, ..., \pi_L$ are the characters of $\mathbf{y}$. Hyper-parameters α, β and γ are weights for the language model, the word penalty, and the silence penalty.

For decoding WSJ, we tune the hyperparameters α, β and γ using a random search. Finally, we decode the emissions from the acoustic model with the best parameter setting for α, β and γ. We use a beam size of 4,000 and beam score threshold of 250 for the word based language models, and a beam size of 1,500

[2]Similar to [16], we found that sampling negatives from different sequences and speakers yields inferior results.

Table 1: *Replacing log-mel filterbanks (Baseline) by pre-trained embeddings improves WSJ performance on test (nov92) and validation (nov93dev) in terms of both LER and WER. We evaluate pre-training on the acoustic data of part of clean and full Librispeech as well as the combination of all of them.* † *indicates results with phoneme-based models.*

			nov93dev		nov92	
			LER	WER	LER	WER
Deep Speech 2 (12K h labeled speech; Amodei et al., 2016)			-	4.42	-	3.1
Trainable frontend [25]			-	6.8	-	3.5
Lattice-free MMI [35]			-	5.66†	-	2.8†
Supervised transfer-learning [36]			-	4.99†	-	2.53†
4-GRAM LM [37]	Pre-Training Data					
Baseline	–	–	3.32	8.57	2.19	5.64
wav2vec	Librispeech	80 h	3.71	9.11	2.17	5.55
wav2vec	Librispeech	960 h	2.85	7.40	1.76	4.57
wav2vec	Librispeech + WSJ	1,041 h	2.91	7.59	1.67	4.61
wav2vec large	Librispeech	960 h	2.73	6.96	1.57	4.32
WORD CONVLM [38]						
Baseline	–	–	2.57	6.27	1.51	3.60
wav2vec	Librispeech	960 h	2.22	5.39	1.25	2.87
wav2vec large	Librispeech	960 h	2.13	5.16	1.02	2.53
CHAR CONVLM [39]						
Baseline	–	–	2.77	6.67	1.53	3.46
wav2vec	Librispeech	960 h	2.14	5.31	1.15	2.78
wav2vec large	Librispeech	960 h	2.11	5.10	0.99	2.43

with beam score threshold 40 for the character based language model.

3.4. Pre-training Models

The pre-training models are implemented in PyTorch in the fairseq toolkit [40]. We optimize them with Adam [41] and a cosine learning rate schedule [42] annealed over 40k update steps for both WSJ and the clean Librispeech training datasets or over 400k steps for full Librispeech. We start with a learning rate of 1×10^{-7}, and then gradually warm it up for 500 updates up to 5×10^{-3} and then decay it following the cosine curve up to 1×10^{-6}. To compute the objective, we sample ten negatives for each of the $K = 12$ tasks.

We train the first wav2vec variant on 8 GPUs and put audio sequences amounting up to 1.5M frames on each GPU. Sequences are grouped by length and we crop each to a maximum size of 150k frames, or the length of the shortest sequence in the batch, whichever is smaller. Cropping removes speech signal from either the beginning or the end of the sequence and we randomly decide the cropping offsets for each sample; we re-sample every epoch. This is a form of data augmentation but also ensures equal length of all sequences on a GPU and removes on average 25 % of the training data. After cropping, the total effective batch size across GPUs is about 556 seconds of speech. For the large model variant, we train on 16 GPUs, doubling the effective batch size.

4. Results

Different to [16], we evaluate the pre-trained representations directly on downstream speech recognition tasks. We measure speech recognition performance on the WSJ benchmark and simulate various low resource setups (§4.1). We also evaluate on the TIMIT phoneme recognition task (§4.2) and ablate various modeling choices (§4.3).

4.1. Pre-training for the WSJ benchmark

We consider pre-training on the audio data (without labels) of WSJ, part of clean Librispeech (about 80 h) and full Librispeech as well as a combination of all datasets (§3.1). For the pre-training experiments we feed the output of the context network to the acoustic model, instead of log-mel filterbank features.

Table 1 shows that pre-training on more data leads to better accuracy on the WSJ benchmark. Pre-trained representations can substantially improve performance over our character-based baseline which is trained on log-mel filterbank features. This shows that pre-training on unlabeled audio data can improve over the best character-based approach, Deep Speech 2 [1], by 0.67 WER on nov92. In comparison to [35], wav2vec performs as well as their phoneme-based model and wav2vec large outperforms it by 0.37 WER. The phoneme-based approach of [36] pre-trains on the labeled version of Librispeech and then fine-tunes on WSJ. wav2vec large still outperforms [36] despite a weaker baseline model and not using Librispeech transcriptions.

What is the impact of pre-trained representations with less transcribed data? In order to get a better understanding of this, we train acoustic models with different amounts of labeled training data and measure accuracy with and without pre-trained representations (log-mel filterbanks). The pre-trained representations are trained on the full Librispeech corpus and we measure accuracy in terms of WER when decoding with a 4-gram language model. Figure 2 shows that pre-training reduces WER by 36 % on nov92 when only about eight hours of transcribed data is available. Pre-training only on the audio data of WSJ (wav2vec WSJ) performs worse compared to the much larger Librispeech (wav2vec Libri). This further confirms that pre-training on more data is important to good performance. Similar to [7], we noticed that fine-tuning the embedding network does not meaningfully improve performance while substantially increasing the acoustic model training time.

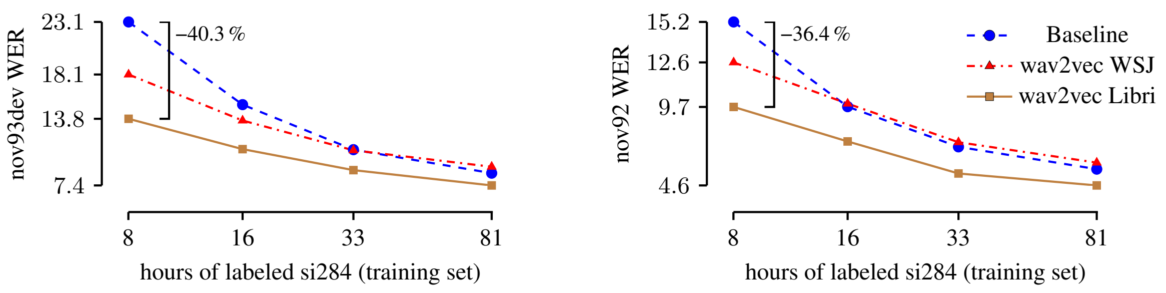

Figure 2: *Pre-training substanstially improves WER in simulated low-resource setups on the audio data of WSJ compared to wav2letter++ with log-mel filterbanks features (Baseline). Pre-training on the audio data of the full 960 h Librispeech dataset (wav2vec Libri) performs better than pre-training on the 81 h WSJ dataset (wav2vec WSJ).*

Table 2: *Results for phoneme recognition on TIMIT in terms of phoneme error rate (PER). All our models use the CNN-8L-PReLU-do0.7 architecture [43].*

	dev PER	test PER
CNN + TD-filterbanks [25]	15.6	18.0
Li-GRU + MFCC [43]	–	16.7 ± 0.26
Li-GRU + FBANK [43]	–	15.8 ± 0.10
Li-GRU + fMLLR [43]	–	14.9 ± 0.27
Baseline	16.9 ± 0.15	17.6 ± 0.11
wav2vec (Librispeech 80h)	15.5 ± 0.03	17.6 ± 0.12
wav2vec (Librispeech)	13.6 ± 0.20	15.6 ± 0.23
wav2vec (Librispeech + WSJ)	**12.9** ± **0.18**	**14.7** ± **0.42**

Table 3: *Effect of different number of negative samples during pre-training for TIMIT on the development set.*

negatives	dev PER	train time (h)
1	16.3	6.1
2	15.8	6.3
5	15.9	8.2
10	**15.5**	10.5
20	15.7	15.3

4.2. Pre-training for TIMIT

On the TIMIT task we use a 7-layer wav2letter++ model with high dropout (§3; [18]). Table 2 shows that wav2vec pre-training on Librispeech and WSJ audio data can lead to results matching the state of the art. Accuracy steadily increases with more data for pre-training and the best accuracy is achieved with the largest amount of data for pre-training.

4.3. Ablations

In this section we analyze some of the design choices we made for wav2vec. We pre-train on the 80 hour subset of clean Librispeech and evaluate on TIMIT. Table 3 shows that increasing the number of negative samples only helps up to ten samples. Thereafter, performance plateaus while training time increases. We suspect that this is because the training signal from the positive samples decreases as the number of negative samples increases. In this experiment, everything is kept equal except for the number of negative samples.

Table 4: *Effect of different crop sizes (left) and different number of tasks (right).*

Crop size	dev PER
None (Avg. 207k)	16.3
100k	16.1
150k	**15.5**
200k	16.0

# Tasks	dev PER
8	15.9
12	**15.5**
16	15.5

Next, we analyze the effect of data augmentation through cropping audio sequences (§3.4). When creating batches we crop sequences to a pre-defined maximum length. Table 4 shows that a crop size of 150k frames results in the best performance. Not restricting the maximum length (None) gives an average sequence length of about 207k frames and results in the worst accuracy. This is most likely because the setting provides the least amount of data augmentation.

Table 4 also shows that predicting more than 12 steps ahead in the future does not result in better performance and increasing the number of steps increases training time.

5. Conclusions

We introduce wav2vec, the first application of unsupervised pre-training to speech recognition with a fully convolutional model. Our approach achieves 2.43 % WER on the test set of WSJ, a result that outperforms the next best known character-based speech recognition model in the literature [1] while using three orders of magnitude less transcribed training data. We show that more data for pre-training improves performance and that this approach not only improves resource-poor setups, but also settings where all WSJ training data is used. In future work, we will investigate different architectures which is likely to further improve performance.

6. Acknowledgements

We thank the Speech team at FAIR, especially Jacob Kahn, Vineel Pratap and Qiantong Xu for help with wav2letter++ experiments, and Tatiana Likhomanenko for providing convolutional language models for our experiments.

7. References

[1] D. Amodei, S. Ananthanarayanan, R. Anubhai, J. Bai, E. Battenberg, C. Case, J. Casper, B. Catanzaro, Q. Cheng, G. Chen *et al.*, "Deep speech 2: End-to-end speech recognition in english and mandarin," in *Proc. of ICML*, 2016.

[2] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *Proc. of CVPR*, 2009.

[3] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft coco: Common objects in context," in *Proc. of ECCV*, 2014.

[4] O. Vinyals, A. Toshev, S. Bengio, and D. Erhan, "Show and tell: Lessons learned from the 2015 MS COCO image captioning challenge," *arXiv*, vol. abs/1609.06647, 2016.

[5] D. Pavllo, C. Feichtenhofer, D. Grangier, and M. Auli, "3d human pose estimation in video with temporal convolutions and semi-supervised training," in *Proc. of CVPR*, 2019.

[6] C. Doersch, A. Gupta, and A. A. Efros, "Unsupervised visual representation learning by context prediction," in *Proc. of ICCV*, 2015.

[7] O. J. Hénaff, A. Razavi, C. Doersch, S. M. A. Eslami, and A. van den Oord, "Data-efficient image recognition with contrastive predictive coding," *arXiv*, vol. abs/1905.09272, 2019.

[8] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," *arXiv*, vol. abs/1810.04805, 2018.

[9] A. Radford, K. Narasimhan, T. Salimans, and I. Sutskever, "Improving language understanding by generative pre-training," https://s3-us-west-2.amazonaws.com/openai-assets/research-covers/language-unsupervised/language_understanding_paper.pdf, 2018.

[10] A. Baevski, S. Edunov, Y. Liu, L. Zettlemoyer, and M. Auli, "Cloze-driven pretraining of self-attention networks," *arXiv*, vol. abs/1903.07785, 2019.

[11] S. Edunov, A. Baevski, and M. Auli, "Pre-trained language model representations for language generation," in *Proc. of NAACL*, 2019.

[12] G. Lample and A. Conneau, "Cross-lingual language model pre-training," *arXiv*, vol. abs/1901.07291, 2019.

[13] Z. Lian, Y. Li, J. Tao, and J. Huang, "Improving speech emotion recognition via transformer-based predictive coding through transfer learning," *arXiv*, vol. abs/1811.07691, 2018.

[14] M. Ravanelli and Y. Bengio, "Learning speaker representations with mutual information," *arXiv*, vol. abs/1812.00271, 2018.

[15] G. Synnaeve and E. Dupoux, "A temporal coherence loss function for learning unsupervised acoustic embeddings," in *Proc. of SLTU*, 2016.

[16] A. van den Oord, Y. Li, and O. Vinyals, "Representation learning with contrastive predictive coding," *arXiv*, vol. abs/1807.03748, 2018.

[17] J. Kunze, L. Kirsch, I. Kurenkov, A. Krug, J. Johannsmeier, and S. Stober, "Transfer learning for speech recognition on a budget," in *Proc. of Workshop on Representation Learning for NLP*, 2017.

[18] G. Synnaeve and E. Dupoux, "A temporal coherence loss function for learning unsupervised acoustic embeddings," *Procedia Computer Science*, vol. 81, pp. 95–100, 2016.

[19] H. Kamper, A. Jansen, and S. Goldwater, "A segmental framework for fully-unsupervised large-vocabulary speech recognition," *Comput. Speech Lang.*, vol. 46, no. C, Nov. 2017.

[20] Y. Chung, W. Weng, S. Tong, and J. R. Glass, "Unsupervised cross-modal alignment of speech and text embedding spaces," *arXiv*, vol. abs/1805.07467, 2018.

[21] Y. Chen, C. Shen, S. Huang, H. Lee, and L. Lee, "Almost-unsupervised speech recognition with close-to-zero resource based on phonetic structures learned from very small unpaired speech and text data," *arXiv*, vol. abs/1810.12566, 2018.

[22] J. Chorowski, R. J. Weiss, S. Bengio, and A. van den Oord, "Unsupervised speech representation learning using wavenet autoencoders," *arXiv*, vol. abs/1901.08810, 2019.

[23] R. Collobert, J. Weston, L. Bottou, M. Karlen, K. Kavukcuoglu, and P. Kuksa, "Natural language processing (almost) from scratch," *JMLR*, 2011.

[24] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," in *Proc. of NIPS*, 2013.

[25] N. Zeghidour, N. Usunier, I. Kokkinos, T. Schaiz, G. Synnaeve, and E. Dupoux, "Learning filterbanks from raw speech for phone recognition," in *Proc. of (ICASSP)*, 2018.

[26] Y. Wu and K. He, "Group normalization," *arXiv*, vol. abs/1803.08494, 2018.

[27] J. S. Garofolo, L. F. Lamel, W. M. Fisher, J. G. Fiscus, D. S. Pallett, and N. L. Dahlgren, "The DARPA TIMIT Acoustic-Phonetic Continuous Speech Corpus CDROM," *Linguistic Data Consortium*, 1993.

[28] J. S. Garofolo, D. Graff, D. Paul, and D. S. Pallett, "CSR-I (WSJ0) Complete LDC93S6A. Web Download," *Linguistic Data Consortium*, 1993.

[29] P. C. Woodland, J. J. Odell, V. Valtchev, and S. J. Young, "Large vocabulary continuous speech recognition using htk," in *Proc. of ICASSP*, 1994.

[30] V. Panayotov, G. Chen, D. Povey, and S. Khudanpur, "Librispeech: an asr corpus based on public domain audio books," in *Proc. of ICASSP*. IEEE, 2015, pp. 5206–5210.

[31] V. Pratap, A. Hannun, Q. Xu, J. Cai, J. Kahn, G. Synnaeve, V. Liptchinsky, and R. Collobert, "wav2letter++: The fastest open-source speech recognition system," *arXiv*, vol. abs/1812.07625, 2018.

[32] R. Collobert, C. Puhrsch, and G. Synnaeve, "Wav2letter: an end-to-end convnet-based speech recognition system," *arXiv*, vol. abs/1609.03193, 2016.

[33] R. Collobert, A. Hannun, and G. Synnaeve, "A fully differentiable beam search decoder," *arXiv*, vol. abs/1902.06022, 2019.

[34] Y. N. Dauphin, A. Fan, M. Auli, and D. Grangier, "Language modeling with gated convolutional networks," in *Proc. of ICML*, 2017.

[35] H. Hadian, H. Sameti1, D. Povey, and S. Khudanpur, "End-to-end speech recognition using lattice-free mmi," in *Proc. of Interspeech*, 2018.

[36] P. Ghahremani, V. Manohar, H. Hadian, D. Povey, and S. Khudanpur, "Investigation of transfer learning for asr using lf-mmi trained neural networks," in *Proc. of ASRU*, 2017.

[37] K. Heafield, I. Pouzyrevsky, J. H. Clark, and P. Koehn, "Scalable modified Kneser-Ney language model estimation," in *Proc. of ACL*, 2013.

[38] N. Zeghidour, Q. Xu, V. Liptchinsky, N. Usunier, G. Synnaeve, and R. Collobert, "Fully convolutional speech recognition," *arXiv*, vol. abs/1812.06864, 2018.

[39] T. Likhomanenko, G. Synnaeve, and R. Collobert, "Who needs words? lexicon-free speech recognition," in *Proc. of Interspeech*, 2019.

[40] M. Ott, S. Edunov, A. Baevski, A. Fan, S. Gross, N. Ng, D. Grangier, and M. Auli, "fairseq: A fast, extensible toolkit for sequence modeling," in *Proc. of NAACL System Demonstrations*, 2019.

[41] D. P. Kingma and J. Ba, "Adam: A Method for Stochastic Optimization," in *Proc. of ICLR*, 2015.

[42] I. Loshchilov and F. Hutter, "SGDR: stochastic gradient descent with restarts," *arXiv*, vol. abs/1608.03983, 2016.

[43] M. Ravanelli, P. Brakel, M. Omologo, and Y. Bengio, "Light gated recurrent units for speech recognition," *IEEE Transactions on Emerging Topics in Computational Intelligence*, vol. 2, no. 2, pp. 92–102, 2018.