Comparative Analysis of the wav2vec 2.0 Feature Extractor

Anonymous submission to INTERSPEECH 2023

Abstract

Automatic speech recognition (ASR) systems typically use handcrafted feature extraction pipelines. To avoid their inherent information loss and to achieve a more consistent modeling from speech to transcribed text, neural raw waveform feature extractors (FEs) are an appealing approach. Also the *wav2vec 2.0* model, which has recently gained large popularity, uses a convolutional FE which operates directly on the speech waveform. However, it is not yet studied extensively in the literature. In this work, we study its capability to replace the standard feature extraction methods in a connectionist temporal classification (CTC) ASR model and compare it to an alternative neural FE. We show that both are competitive with traditional FEs on the LibriSpeech benchmark and analyze the effect of the individual components. Furthermore, we analyze the learned filters and show that the most important information for the ASR system is obtained by a set of bandpass filters.

Index Terms: speech recognition, feature extraction, raw waveform modeling, wav2vec 2.0

1. Introduction

acoustic models (AMs) for automatic speech recognition (ASR) traditionally makes use of hand-designed feature extraction methods like log Mel filterbank features or Gammatone features [1]. These techniques are motivated by insights into the properties of the human auditory system. However, their fixed design inevitably leads to information loss during the feature extraction.

This issue can be addressed by neural feature extractors (FEs). Since their parameters are learned during training, these FEs can adapt to the need of AMs and the loss of particularly helpful information may be avoided. A number of works have been proposed in this direction, often using convolutional neural networks (NNs) [2, 3, 4] but also applying parametrized functions [5] and other architectures [6].

Recently, the *wav2vec 2.0* framework [7] has received large attention. It is mainly known for its contrastive self-supervised pre-training, which allows to pre-train a model on large amounts of unlabeled audio data. Subsequently, it can be fine-tuned for a specific downstream task at hand. While the context network with multiple *Transformer* blocks constitutes the major part of the architecture, the *wav2vec 2.0* model also makes use of a convolutional feature encoder. Feature encoder [7, 8] and feature extractor [9, 10] are both common terms in the literature referring to the same part of the model and for consistency, we only use the term feature extractor (FE) here as it is more in line with other works on neural front-ends.

While the *wav2vec 2.0* FE is similar to other neural feature extraction methods, it is not yet closely studied in the literature. [11] studies the FE by feeding synthesized sine waves. Other works that analyze *wav2vec 2.0* models rather focus on the large *Transformer* part of the network which operates at a higher level of abstraction [12, 13, 14, 15].

In this work, we show new insights into the *wav2vec 2.0* FE. Unlike previous work, we demonstrate that it can be used as a replacement for traditional feature extraction methods for a connectionist temporal classification (CTC) model similar to supervised convolutional (SC) features. The results show that both are competitive on the LibriSpeech benchmark. We analyze the learned filters and show the differences between the characteristics of both neural FEs. Finally, we demonstrate that the learned bandpass filters of the SC FE encode the valuable information for the ASR system while the remaining wideband filter information is largely ignored.

2. Related work

This paper takes up the line of research that applies learnable feature extraction front-ends for ASR [2, 3, 4, 5, 6]. In particular, different neural FEs are compared to traditional feature extraction methods. We choose to investigate the supervised convolutional FE from [4], that was also studied in [16]. Furthermore, the feature extractor of *wav2vec 2.0* [7] is used as a learnable front-end for supervised ASR training.

The *wav2vec* framework has been established in a series of related papers [17, 18, 7, 8]. Besides modifications of the self-supervised training criteria – moving from the future time step prediction in [17] to masked time step prediction – and incorporating quantization modules, also the architecture has been revised. While [17] uses a fully convolutional architecture, [7] uses a large stack of *Transformer* blocks to contextualize the convolutional representations. Also the FE itself has been adjusted. [17] uses 5 layers, a group normalization with a single group and a ReLU nonlinearity. In contrast, 7 layers with group normalization in the first layer and a GELU activation function are used in [7]. Additionally, layer normalization and a linear projection are added after the convolutional layers.

While most works analyzing the *wav2vec 2.0* model focus on the *Transformer* part, there are papers which study the FE as well. The authors of [11] feed synthesized sine waves to a pre-trained model and observe that enough temporal detail is obtained and information about fundamental frequencies is represented in the model's output. In particular, they claim that the frequencies are distributed linearly over the Hertz scale, unlike perceptually motivated scales like the Mel scale or the Greenwood function for Gammatone features. [12] states that the learned representations of the FE are highly correlated with Mel spectrogram features as was determined by the canonical correlation analysis (CCA), where the correlation is at a high plateau for FE layers 3 to 7. This observation is supported by experimental evidence in [15].

[9] investigates different settings for the *wav2vec 2.0* FE and shows that the required computation can be significantly reduced while maintaining similar performance. Furthermore, reducing the FE's size in favor of a larger context network shows promising results. In addition, a deeper variant with additional point-wise convolutional layers is proposed which outperforms its wider counterpart using higher inner dimensions. However, unlike our work this is always done in the whole *wav2vec 2.0* framework including pre-training and fine-tuning and no insights into the learned representations are given.

3. Methods

This work compares different FEs' applicability for ASR. We always normalize the waveform to zero mean and unit variance before it is input to a FE. The FE is followed by the remaining AM. While the separation between FE and remaining AM is not sharp for neural feature extraction, we always refer to the part that replaces the traditional feature extraction as FE.

3.1. Feature Extractors

3.1.1. Standard Feature Extraction

As a reference, we use standard, hand-designed FEs– namely log Mel filterbank features and Gammatone features. To compute the log Mel filterbank features, first the short time Fourier transform (STFT) of the waveform is computed with a window size of $25\,\mathrm{ms}$ and a shift of $10\,\mathrm{ms}$. We keep the square of the magnitude and perform Mel warping to obtain a 80-dim vector. Finally, $\log_{10}$ and normalization are applied.

Gammatone features apply a Gammatone filterbank to the pre-emphasized speech signal, perform temporal integration over each channel using a low pass filter, i.e., the Hanning window with a size of $25\,\mathrm{ms}$ and a shift of $10\,\mathrm{ms}$, compress using the 10^{th} root and finally compute the discrete cosine transform (DCT) of the values [1]. Further, the the resulting 50-dim features are normalized.

3.1.2. wav2vec 2.0 Feature Extractor

The FE of the *wav2vec 2.0* model [7] mainly consists of 7 convolutional layers. They are configured with kernel sizes (10, 3, 3, 3, 3, 2, 2), strides (5, 2, 2, 2, 2, 2, 2), and a GELU activation function. The first layer applies group normalization and finally, layer normalization and a linear projection is added. The total receptive field of the *wav2vec 2.0* FE is $25\,\mathrm{ms}$ and the shift between consecutive frames is $20\,\mathrm{ms}$. Since the other FEs operate at a shift of $10\,\mathrm{ms}$ and the *wav2vec 2.0* FE is built in a modular way, we remove the last layer with a stride of 2 in order to achieve features at the same frame rate. This reduces the total receptive field to $15\,\mathrm{ms}$.

3.1.3. Supervised Convolutional Features

As an alternative comparable neural feature extraction method, we use the SC features from [4, 16]. Similarly, a convolutional filterbank with learnable parameters is applied to the waveform. As in the case of Gammatones, a temporal integration is performed over each channel. However, in this case multiple filters are used for temporal integration allowing for multi-resolutional processing. Additionally, these filters are learned during training. By default, we set the size of the first filterbank to 160, its strides to 10 and the number of channels to 150. The second convolutional filter applies 5 temporal integration filters with a size of 40 and strides of 16 each. Since the output of all 5 filters is stacked, we have a resulting feature dimension of 750. The total receptive field of the SC features is $40\,\mathrm{ms}$.

3.2. Acoustic Model

The AM mainly consists of 3 VGG blocks for downsampling and 12 *Conformer* blocks. Its configuration mostly follows [19] and [20]. The VGG downsampling uses three 3x3-convolutional layers with 32, 64 and 64 channels, respectively. In total, a subsampling factor of 4 in the time dimension is achieved by the strided convolutions so that the *Conformer* operates on frames with a shift of $40\,\mathrm{ms}$. The feature dimension is reduced by a factor of 2 due to max pooling and then multiplied by the number of channels in the last convolution resulting in a total increase by factor 32 in our case. After a linear projection to a 512-dimensional representation and dropout of 0.1, the 12 *Conformer* blocks with an inner dimension of 512, swapped convolution and multi-headed self-attention modules and 8 heads are applied.

3.3. Connectionist Temporal Classification

We use a CTC model for our experiments. It follows the setup described for the CTC model in [20]. All models are trained in a purely supervised fashion using the CTC objective unless explicitly stated otherwise. The official LibriSpeech 4-gram language model (LM) is used and integrated using shallow fusion [21] during decoding.

4. Experimental Setup

4.1. Data

We conduct experiments on the 960h LibriSpeech corpus [22], which consists of read English speech. For the experiments using a pre-trained *wav2vec 2.0* FE, the same 960h were used in pre-training. The LM was trained on the default LibriSpeech LM training data which includes the transcriptions of the training set along with the additional 800M-word text only data. The evaluation is done on the standard *dev-clean*, *dev-other*, *test-clean* and *test-other* sets.

4.2. Training Details

We generally follow the setup in [20]. The batch size is 640k samples, which corresponds to $40\,\mathrm{s}$ of speech waveform and the gradients are accumulated over 3 steps. The NAdam optimizer uses a one-cycle learning rate (OCLR) schedule with a peak learning rate of $3 \cdot 10^{-4}$ and we train the model for 20 epochs. In contrast to [20], no gradient noise is applied. Each model is trained on a single graphics processing unit (GPU).

5. Results

The results, namely the word error rates (WERs) on the LibriSpeech *dev* and *test* sets, for the different FEs are shown in Table 1. The baseline performance using Gammatone features is improved compared to [20] mainly by disabling gradient noise and re-tuning the peak learning rate. We can observe that all FEs are within a close range. The SC features slightly outperform the Gammatone baseline, while log Mel features and the *wav2vec 2.0* FE are again better with advantages on *dev-other* and *test-other*, respectively. This demonstrates that learnable neural FEs are competitive with hand-designed methods in a purely supervised setup with 960h of training data. However, a clear advantage of these techniques cannot be deduced from the experimental results.

The number of parameters for Gammatone and log Mel features refers to the finite impulse response (FIR) and Mel filterbanks, respectively, that we use in our implementation even if they are not trainable. The differences in the total number of parameters are influenced by the size of the FE, but also of the linear layer before the *Conformer* which grows large as the feature dimension increases.

Table 1: *Comparison of different feature extraction methods for a CTC model on LibriSpeech.*

Feature Extractor		#Params		WER [%]			
				dev		test	
Type	Name	Total	FE	clean	other	clean	other
Fixed	Gammatone	73.7M	32k	3.0	7.1	3.4	7.7
	Mel Filterbank	74.2M	21k	2.9	6.9	3.3	7.2
NN	SC	85.2M	26k	2.9	7.0	3.4	7.6
	wav2vec 2.0	89.5M	4.1M	2.9	6.8	3.3	7.5

5.1. Dissecting the wav2vec 2.0 Feature Extractor

It is striking that the *wav2vec 2.0* FE uses two orders of magnitude more parameters than the SC architecture. To understand where *wav2vec 2.0* uses the them and how much they contribute to the FE's performance, we run it with different configurations. The results for different widths and depths are shown in Table 2. The kernel sizes and strides were chosen as (10, 3, 3, 3, 3, 2), (5, 2, 2, 2, 2, 2) for 6 layers, (10, 6, 3, 3, 3), (5, 4, 2, 2, 2) for 5 layers, (10, 6, 6, 3), (5, 4, 4, 2) for 4 layers, (20, 6, 6), (10, 4, 4) for 3 layers and (32, 20), (16, 10) for 2 layers. The results show that decreasing the inner dimension deteriorates the performance, however, a larger dimension does not improve over the 512-dim baseline. In contrast, no major effect can be observed regarding the number of layers. We also tried a variant with only one layer using kernel size 320 and stride 160 which did not converge. Removing the final projection with about 394k parameters has no significant impact.

The proposed FE in [9] uses increasing inner dimensions and we use this approach in the last three lines of Table 2. The first layer uses a dimension of 64/128 and it is doubled after each layer with an odd index. Additionally, we add point-wise convolutional layers after all but the first layer using the same inner dimension as the preceding layer for the last row of the table. It can be observed that these configurations perform similarly to the 6 layer 512-dim baseline while reducing the training time by about 21%, 19% and 14%, respectively. Note that the SC features perform similar to the variant with dimension 64-512 with again around 10% shorter training time.

5.2. *wav2vec 2.0* Pre-Training

The main appeal of the *wav2vec 2.0* framework is the ability to pre-train the model on unlabeled audio data. [7] even reports relative WER improvements of about 10% on the *other* subsets for their experiments on LibriSpeech when pre-training on the same data as used in supervised training compared to pure supervised training from scratch. Here, we study this effect only for the FE by using the parameters of an FE pre-trained on LibriSpeech[1], so no external data is added. During the supervised training, the FE weights can be further trained or kept frozen. As shown in Table 3, no positive effect can be observed here. In fact, using a frozen pre-trained FE shows clearly worse performance than training the FE purely supervised from scratch. We hypothesize that the pre-training mainly benefits the *Transformer* encoder and not the FE while the mismatch between the *Transformer* encoder in pre-training and *Conformer* encoder in our setup could also pose a challenge for the case that the FE is frozen.

[1] `https://github.com/facebookresearch/fairseq/blob/main/examples/wav2vec/README.md`

Table 2: *Studying the effect of the wav2vec 2.0 feature extractor's width and depth.*

#Layers	Dim	#Params FE	WER [%]			
			dev		test	
			clean	other	clean	other
6	1024	15.5M	2.9	6.8	3.3	7.5
	512	4.1M	2.9	6.8	3.3	7.5
	256	1.1M	2.9	7.1	3.4	7.6
	128	330k	3.0	7.2	3.4	7.7
	64	108k	3.1	7.4	3.5	7.8
5	512	4.3M	2.9	7.1	3.4	7.6
	64	112k	3.0	7.4	3.5	7.9
4	512	4.3M	2.9	7.0	3.3	7.6
	64	112k	3.1	7.2	3.4	7.8
3	512	3.6M	2.9	6.9	3.3	7.4
	64	101k	3.1	7.4	3.5	7.9
2	512	5.6M	3.0	7.1	3.5	7.7
	64	134k	3.1	7.6	3.4	8.2
6	64-512	1.0M	3.0	7.1	3.3	7.7
	128-1024	3.3M	2.9	7.0	3.3	7.5
11		5.0M	2.9	6.9	3.3	7.4

Table 3: *Effect of pre-training the wav2vec 2.0 feature extractor using the unsupervised loss on the same data (LibriSpeech 960h).*

Feature Extractor		WER [%]			
		dev		test	
Trainable	Pre-trained	clean	other	clean	other
yes	no	2.9	6.8	3.3	7.5
	yes	2.9	6.9	3.3	7.5
no		3.0	7.1	3.3	7.8

5.3. Frequency Response of Learned Filters

To analyze the learned filters of the neural FEs, we first plot the frequency response of the filters in the first layer which operates on the raw waveform. As the order of learned filters in a convolutional layer is arbitrary, we sort the filters by the peak value of the frequency response and by the upper and lower 3 dB cutoff frequencies as second and third criteria. The plots are depicted in Figure 1 (a)-(d), which share the same color range.

The well known distribution of the Gammatone filters is given in Figure 1 (a) as a reference. Figure 1 (b) shows the frequency response of the learned SC filters. In line with [4], bandpass filters with a non-linear distribution of center frequencies are learned. This confirms, that these filters are also preferred by more advanced neural back-ends like the self-attention-based *Conformer* which are superior in the exploitation of context information compared to simpler neural back-ends e.g. in [23]. The distribution resembles the Gammatone filterbank and with

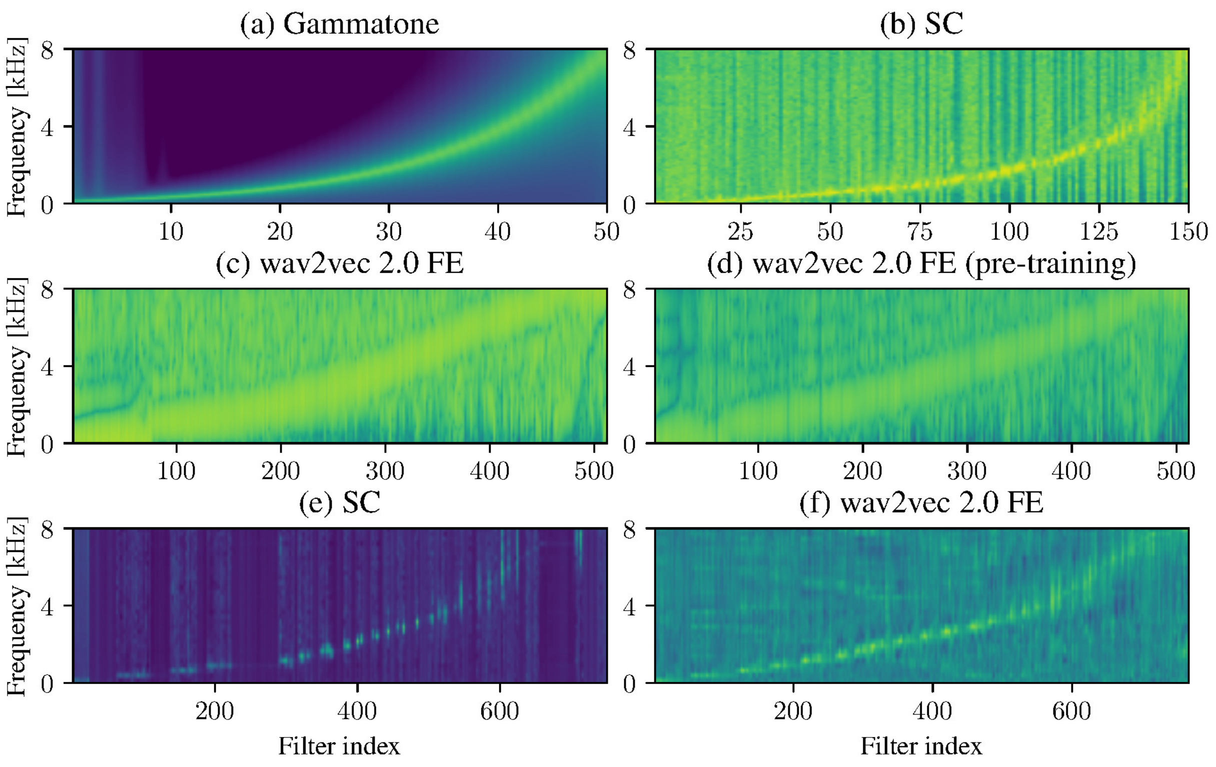

Figure 1: *Frequency responses for (parts of) the different FEs. (a)-(d) show the frequency response of the filters in the first layer that operates on the waveform. (e)-(f) show the frequency response of the complete learned FE.*

higher center frequencies, the passband is also wider. However, the stopband attenuation is clearly weaker and there is also a number of filters with no clear passband and a rather uncharacteristic frequency response.

The distribution for the *wav2vec 2.0* FE's first layer in Figure 1 (c) does not entail this degree of resemblance. Again, bandpass filters are learned, however, they have a wider passband, their center frequencies are distributed rather linearly and the stopband attenuation is weak. In addition, about 14% of all filters have their peak at 0 kHz and another 10% at 8 kHz with varying bandwidths. Moreover, a number of filters exists with multiple passbands. Interestingly, the frequency response of the pre-trained *wav2vec 2.0* FE in Figure 1 (d) is very similar to Figure 1 (c). This indicates that the properties learned in unsupervised pre-training and supervised CTC training are similar.

Figure 1 (e) and (f) show the frequency response of the complete FE. For *wav2vec 2.0*, the distribution of center frequencies is not as linear anymore and as suggested in [11]. Moreover, the ratio of peak to mean values is much lower than for the SC features.

5.4. Filter Masking

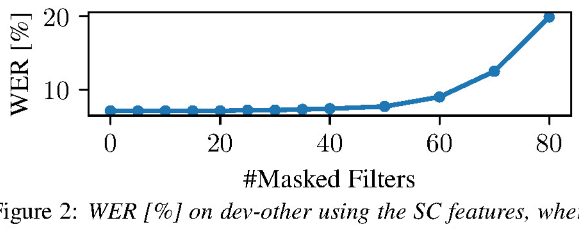

Figure 2: *WER [%] on dev-other using the SC features, where N out of 150 filters of the first layer are masked in recognition based on the peak-to-average ratio of their frequency response.*

The observed uncharacteristic wideband filters from Section 5.3 raise the question whether their output is particularly helpful for the model in addition to the bandpass filters or if they are just not trained properly and ignored by the system. To answer this question, we masked a portion of the 150 filters in the SC features' first layer. The filters are sorted by the peak-to-average ratio of their frequency response and the N filters with the lowest ratio are masked out during recognition. In Figure 2, we can observe that masking the first 20 filters has no effect on the WER and only after masking 50, so a third of all filters, the WER degrades significantly. This demonstrates that the wideband filters do not contain valuable information for the ASR system and are rather the result of a suboptimal training process. Future work should address this observation to allow a better exploitation of the FE's potential.

6. Conclusions

In this work, we utilize the *wav2vec 2.0* FE as a replacement for traditional feature extraction methods in a CTC ASR model on LibriSpeech. It shows competitive performance and outperforms an alternative neural FE, namely the SC features, slightly. We demonstrate that its width is more responsible for its capabilities that its depth and show that pre-training the FE only is not beneficial. Finally, we analyze the learned filters and highlight the differences between both neural FEs. In particular, it is shown that the SC FE's learned bandpass filters encode the valuable information for the ASR system while the remaining wideband filter information is largely ignored.

7. Acknowledgements

The acknowledgements would likely reveal the author identity and are therefore hidden for the double-blind review process.

8. References

[1] R. Schlüter, I. Bezrukov, H. Wagner, and H. Ney, "Gammatone features and feature combination for large vocabulary speech recognition," in *Proc. ICASSP*, Honolulu, HI, USA, Apr. 2007, pp. 649–652.

[2] D. Palaz, M. M. Doss, and R. Collobert, "Convolutional neural networks-based continuous speech recognition using raw speech signal," in *Proc. ICASSP*, Brisbane, Australia, Apr. 2015, pp. 4295–4299.

[3] P. Golik, Z. Tüske, R. Schlüter, and H. Ney, "Convolutional Neural Networks for Acoustic Modeling of Raw Time Signal in LVCSR," in *Proc. Interspeech*, Dresden, Germany, Sep. 2015, pp. 26–30.

[4] Z. Tüske, R. Schlüter, and H. Ney, "Acoustic modeling of speech waveform based on multi-resolution, neural network signal processing," in *Proc. ICASSP*, Calgary, Canada, Apr. 2018, pp. 4859–4863.

[5] M. Ravanelli and Y. Bengio, "Speaker recognition from raw waveform with SincNet," in *Proc. SLT*, Athens, Greece, Dec. 2018, pp. 1021–1028.

[6] T. N. Sainath, O. Vinyals, A. Senior, and H. Sak, "Convolutional, long short-term memory, fully connected deep neural networks," in *Proc. ICASSP*. Brisbane, Australia: IEEE, Apr. 2015, pp. 4580–4584.

[7] A. Baevski, Y. Zhou, A. Mohamed, and M. Auli, "wav2vec 2.0: A framework for self-supervised learning of speech representations," vol. 33, 2020, pp. 12 449–12 460.

[8] A. Conneau, A. Baevski, R. Collobert, A. Mohamed, and M. Auli, "Unsupervised cross-lingual representation learning for speech recognition," in *Proc. Interspeech*, Brno, Czechia, 2021, pp. 2426–2430.

[9] F. Wu, K. Kim, J. Pan, K. J. Han, K. Q. Weinberger, and Y. Artzi, "Performance-efficiency trade-offs in unsupervised pre-training for speech recognition," in *Proc. ICASSP*. Singapore: IEEE, May 2022, pp. 7667–7671.

[10] A. Vyas, W. Hsu, M. Auli, and A. Baevski, "On-demand compute reduction with stochastic wav2vec 2.0," in *Proc. Interspeech*, Incheon, Korea, Sep. 2022, pp. 3048–3052.

[11] K. Choi and E. J. Yeo, "Opening the black box of wav2vec feature encoder," *arXiv preprint arXiv:2210.15386*, 2022.

[12] A. Pasad, J.-C. Chou, and K. Livescu, "Layer-wise analysis of a self-supervised speech representation model," in *Proc. ASRU*. IEEE, 2021, pp. 914–921.

[13] Z. Fan, M. Li, S. Zhou, and B. Xu, "Exploring wav2vec 2.0 on speaker verification and language identification," in *Proc. Interspeech*, Brno, Czechia, 2021, pp. 1509–1513.

[14] Y. Li, Y. Mohamied, P. Bell, and C. Lai, "Exploration of a self-supervised speech model: A study on emotional corpora," in *Proc. SLT*. Doha, Qatar: IEEE, Jan. 2023, pp. 868–875.

[15] T. tom Dieck, P.-A. Pérez-Toro, T. Arias-Vergara, E. Nöth, and P. Klumpp, "Wav2vec behind the scenes: How end2end models learn phonetics," in *Proc. Interspeech*, Incheon, Korea, Sep. 2022, pp. 5130–5134.

[16] P. Vieting, C. Lüscher, W. Michel, R. Schlüter, and H. Ney, "On architectures and training for raw waveform feature extraction in ASR," in *Proc. ASRU*. IEEE, 2021, pp. 267–274.

[17] S. Schneider, A. Baevski, R. Collobert, and M. Auli, "wav2vec: Unsupervised pre-training for speech recognition," in *Proc. Interspeech*, Graz, Austria, Sep. 2019, pp. 3465–3469.

[18] A. Baevski, S. Schneider, and M. Auli, "vq-wav2vec: Self-supervised learning of discrete speech representations," in *International Conference on Learning Representations*, Sep. 2019.

[19] M. Zeineldeen, J. Xu, C. Lüscher, W. Michel, A. Gerstenberger, R. Schlüter, and H. Ney, "Conformer-based hybrid ASR system for switchboard dataset," in *Proc. ICASSP*. Singapore: IEEE, May 2022, pp. 7437–7441.

[20] W. Zhou, W. Michel, R. Schlüter, and H. Ney, "Efficient training of neural transducer for speech recognition," pp. 2058–2062, Sep. 2022.

[21] C. Gulcehre, O. Firat, K. Xu, K. Cho, L. Barrault, H.-C. Lin, F. Bougares, H. Schwenk, and Y. Bengio, "On using monolingual corpora in neural machine translation," *arXiv preprint arXiv:1503.03535*, 2015.

[22] V. Panayotov, G. Chen, D. Povey, and S. Khudanpur, "LibriSpeech: an ASR corpus based on public domain audio books," in *Proc. ICASSP*, South Brisbane, Queensland, Australia, Apr. 2015, pp. 5206–5210.

[23] Z. Tüske, P. Golik, R. Schlüter, and H. Ney, "Acoustic modeling with deep neural networks using raw time signal for LVCSR," in *Proc. Interspeech*, Singapore, Sep. 2014, pp. 890–894, ISCA Best Student Paper Award.