INTERSPEECH 2023
20-24 August 2023, Dublin, Ireland

On the (In)Efficiency of Acoustic Feature Extractors for Self-Supervised Speech Representation Learning

Titouan Parcollet, Shucong Zhang, Alberto Gil C. P. Ramos, Rogier van Dalen, Sourav Bhattacharya

Samsung AI Center, Cambridge, United-Kingdom

t.parcollet@samsung.com

Abstract

Speech representations learned with self-supervised learning (SSL) have the potential to significantly improve the performance of a number of audio applications, especially when availability of labeled data from the deployment domain is limited. Despite their successes, SSL training methods are compute- and memory-heavy, and require large investments in computing infrastructure, thus putting it out of the reach of most institutions. Therefore, building efficient model architectures is essential for the wide-scale adoption of SSL in speech technologies. CNN-based Acoustic Feature Extractors (AFE), which are widely used as encoders of acoustic waveforms, remain one of the main efficiency bottlenecks. This work proposes replacing CNN-based AFEs with more efficient ones and demonstrates that SSL pre-training time and memory consumption can be reduced by a factor of two to three over existing methods while preserving performances in speech-, command-, and speaker-recognition tasks.

Index Terms: Efficient speech representation learning, SSL.

1. Introduction

Self-Supervised Learning (SSL) of deep learning systems leverages a vast amount of unlabeled data to deliver groundbreaking performance across a wide range of domains, including computer vision, robotics [1, 2], audio, speech, and language processing [3, 4]. Especially most sub-fields of speech processing increasingly use large pre-trained SSL models to reach previously unseen performance. For instance, SSL has led to state-of-the-art (SOTA) performance for Automatic Speech Recognition (ASR) [5, 6], automatic emotion recognition [7, 8], automatic speaker verification [9, 8], Automatic Speech Translation (AST) [10, 11], Spoken Language Understanding (SLU) [8, 12], speech enhancement [13, 14], and speech separation [13, 15]. Alongside remarkable performance in English, SSL-trained models are reducing the gap between low- and high-resource languages [11, 10], increasing the accessibility for cutting-edge technologies across many languages.

However, SSL pre-training requires significant computing infrastructure limiting participation to a handful of industrial actors. For example, pre-training of a SOTA SSL architecture requires extremely large datasets, e.g., tens or hundreds of thousand hours of speech. It also requires architectures with billions of neural network parameters to reach an optimal level of performance, so that a single training can require 64 or 128 high-end GPUs to complete a single pre-training process and take a few weeks [7]. This leads to high environmental costs and can easily run into hundreds of thousands of US dollars. The high cost alone acts as a high barrier for accessibility in the development of novel SSL techniques. Current SSL leaderboards, such as the SUPERB Benchmark, are occupied by models originating from two companies only [15].

It is therefore vital to make SSL training more efficient. Three culprits for the high cost can be identified [16]: (i) the Acoustic Feature Extractor (AFE), which transforms the raw waveform into a latent representation; (ii) the "context encoder", which is often a large Transformer; and (iii) the SSL training objective. The last aspect, the objective, has seen efficiency improvements such as Data2Vec [17] and HuBERT [5]. Improvements to the first two parts, on the other hands, are hardly explored. SOTA SSL-trained models rely on the same CNN-based AFE combined with a large vanilla Transformer. Careful engineering of these two parts could lead to significant efficiency gains [18] and allow pre-training on mid-tier GPU (e.g., Nvidia Ti 80/90 families) [16].

There exists some scattered work on improving the efficiency of the AFE. One line of research, which this paper will continue, is to exploit decades of research in the signal processing domain aiming at extracting the best representation of the speech signal. [19] proposed to train a small HuBERT model with *Mel filterbanks* instead of the standard CNN-based AFE. However, the paper focusses on downstream performance and the streaming capabilities of the model, and without mention of pre-training efficiency. [20], on the other hand, introduced a new CNN-based AFE inspired by the wav2vec 2.0 AFE [21], but optimized to increase throughput. This demonstrates that a careful design combined with a time decimation of the input sequence could lead to improvements both in training time and in downstream performance. However, there exists no systematic study of the gains in efficiency and accessibility from choosing different AFEs on speech SSL pre-training. This paper has three main contributions:

1. Introduce new efficient AFEs for training with speech SSL.
2. A systematic evaluation of different efficiency metrics over eight AFEs on a carefully crafted contrastive SSL pre-training task with mid-tier GPUs (e.g. RTX 3090).
3. An examination of the effects on downstream performance for Automatic Speech Recognition (ASR), Keyword Spotting (KS), and Automatic Speaker Verification (ASV).

The conducted experiments, implemented with methods from the widely-adopted SpeechBrain toolkit [22] to facilitate reproducibility, show that a few AFEs introduced in this article make pre-training faster, e.g., reducing from 7 days for a wav2vec 2.0 to 1.8 days (Section 3.1), as well as a more memory efficient, e.g., from 80 GB to 24 GB GPU (Section 3.2), with no degradation in downstream performance. Hence, greater accessibility to speech SSL research can be reached without sacrificing task-specific results or high implementation complexity.

10.21437/Interspeech.2023-1510

2. Acoustic Feature Extractors

In this section, we consider AFEs that can replace the standard and inefficient CNN AFE [21] in wav2vec 2.0 to improve the efficiency of pre-training. In section 2.1, we introduce the AFE in standard wav2vec 2.0 and a more efficient variance, SEW. In section 2.2, we consider AFEs based on Mel filterbanks and learnable filters, with fewer trainable parameters.

2.1. Existing AFEs used in self-supervised learning

The majority of large-scale speech SSL models rely on a one-dimensional and end-to-end convolutional network first introduced in wav2vec 2.0 [21].

Wav2vec 2.0. The original wav2vec 2.0 AFE, also used in WavLM [6] and HuBERT [5], is composed of a number of 1D-convolutions operating directly on the raw audio waveform. The direct connection to the high-dimensional waveform has been identified as the main reason for the high VRAM consumption of this AFE [23, 20]. We use the original wav2vec 2.0 frontend [21], which has seven 1D-convolutional layers with kernels and strides equal to $[11, 3, 3, 3, 3, 3, 3]$ and $[5, 2, 2, 2, 2, 2, 2]$. This is equivalent to a 25 ms window, with a hop length of 20 ms, resulting in an output frequency of 49 Hz (i.e. 49 vectors emitted per second of speech).

SEW. The "Squeezed and Efficient Wav2vec" (SEW) [20] AFE shares a similar network architecture as the wav2vec 2.0 AFE, with almost identical receptive field, e.g., window of 24.8 ms and the hop length of 20 ms. Compared to the wav2vec 2.0 AFE, the SEW AFE reduces the computational complexity by cutting the number of channels in the lower layers as well as the kernel size. The AFE contains seven 1D-convolutional layers with kernel sizes $[7, 3, 3, 3, 3, 2, 2]$ and stride $[5, 2, 2, 2, 2, 2, 2]$ respectively. The AFE doubles the number of channels whenever the sequence length is reduced by a factor of four generating channels size of $[64, 128, 128, 256, 256, 512, 512]$.

2.2. AFEs inspired by Speech Processing

Representing the speech signal in an efficient way has been an active field of research well before the emergence of deep learning, offering plenty of techniques [24, 25]. We would like to exploit this knowledge to build a more efficient AFE for training with SSL.

Mel Filterbanks. Mel filterbanks, whose outputs we will call "FBanks", are a well-known and extensively used transformation of the waveform to the frequency domain inspired by human hearing [26]. Even large pre-trained models, such as Whisper [27], rely on FBank as an input representation. FBank are extremely quick and cheap to compute enabling a good compression of any speech waveform at a very low cost. A standard and common parametrization of FBank gives a window of size 25 ms and a hop length of 10 ms with a per-frame size of 40 or 80 bins. Lin et al. [19] have first proposed to pre-train a small HuBERT model using FBanks only. However, the authors do not explore the efficiency and downstream superiority of their approach. In this paper, 80 FBanks are extracted every 20 ms and with a window size of 25 ms to mimic the time resolution of the wav2vec 2.0 AFE, resulting in an output frequency of 50 Hz.

FastAudio. This AFE is a learnable FBank acoustic extractor. Speech filters are trained alongside the rest of the neural network to optimise the considered objective [28]. For instance, in FastAudio, triangular FBanks are initialized following the standard mel-scale, before adapting their central frequencies and frequency bands during training. FastAudio is not faster than a standard Fbank extraction at training time, and it only becomes equivalent at inference. However, FastAudio has never been trained with SSL and unlike the original FastAudio implementation, we propose to only optimize the central frequencies alongside the rest of the neural network. Indeed, adding the bands as learnable parameters would make FastAudio filters collapse to a single value while combined with the quantization process of wav2vec 2.0 pre-training. Fixing the bands forces FastAudio to keep filtering the signal properly. The output frequency is equivalent to FBanks.

Mel Filterbanks and CNN. FBanks have been combined for years with 1D or 2D CNNs to either enhance their feature representation or reduce their time resolution. For instance, most SOTA ASR systems using Transformers, Transducers, or recurrent architectures, rely on this type of AFEs. To the best of our knowledge, these AFEs have never been properly investigated in the context of speech SSL, and this paper proposes two different combinations of FBanks and CNN following common approaches from the speech-processing literature. First, *"FBank-CNN1d"* aggregates a standard mel-filterbanks extraction from a 25 ms window and a hop length of 10 ms with a two-layered one-dimensional CNN made of 512 filters with kernel sizes and strides of $[3, 3]$, $[2, 1]$ respectively. Layer-normalization and GeLU activations are applied between each layer. The resulting output frequency is 50 Hz. Second, *"FBank-CNN2d"* is inspired by SOTA Transformer architectures originating from well-known toolkits. Hence, it combines the same standard mel-filterbanks parametrization with a two-layered two-dimensional convolutional CNN made of $(128, 64)$ filters and kernel sizes and strides of $[3, 3]$, $[2, 2]$ respectively. The output frequency is therefore halved to 25 Hz, while the feature dimension is increased from 512 for *"FBank-CNN1d"* to 1280. Layer normalizations and Leaky ReLU activations are applied between each layers.

SincNet and Leaf Following the success of mel-filterbanks and CNN, Leaf [29], and SincNet [30] merge those two techniques into a single paradigm. Here, the mel-filterbank computation is replaced with a one-dimensional convolutional layer parametrized with common signal processing filters, e.g., triangular for SincNet and Gabor filters for Leaf. The parameters of those filters are then optimized via backpropagation with the task of interest. Other convolutional layers are then added to the filtering layer to complete the AFE. It results in layers with lower computational and memory costs than CNN layers as they require fewer parameters. SincNet and Leaf AFEs have never been applied to speech SSL before. Based on previous experiments for end-to-end ASR [30], we parametrize the SincNet layer with filters of size 129 with a stride of 5 samples. The number of filters is reduced from 512 in [30] to 128 to further reducing the complexity of the CNN. The convolutional front end is composed of five one-dimensional layers of $[128, 128, 256, 256, 512]$ filters with kernel sizes and strides of $[3, 3, 3, 3, 3]$, $[2, 2, 2, 2, 2]$ respectively. The resulting output frequency is equal to 50 Hz. Lastly, Leaf follows its original implementation [29] adapted to the 20 ms hop-length of the wav2vec 2.0 to get an output frequency of 50 Hz.

3. Experiments

In this section we present the performance of each AFE following a two-step process. First, the efficiency is benchmarked to offer empirical insights on potential training speed and memory gains (see §3.1). Then, downstream performance is evaluated, offering a complete view of the advantages of each solution, and following three tasks from the SUPERB benchmark [15]: automatic speech recognition (ASR), automatic speaker verification (ASV), and keyword spotting (KS) (see §3.2).

Datasets. The Librispeech dataset is used to evaluate both efficiency and downstream performance. During the former, two different sets are created by randomly selecting 1,000 sentences from the 100 hours clean set and then reducing them to either 5 s or 15s. For full SSL pre-training, however, the full Librispeech dataset of 960 hours is considered, with an upper duration limit of 30 s to maximise downstream performance.

Practical considerations. Due to the high pre-training cost associated with contrastive learning, no architecture search has been conducted. We finalized learnable AFE architectures by running a few pre-training epochs with a small set of architectural hyperparameters taken from the literature. In all experiments, AFEs are the only difference between all models and the rest of the architecture strictly follows the original wav2vec 2.0 *BASE* model [21], as described in the SpeechBrain Librispeech recipe (commit *71c1490*). Efficiency experiments are powered with an isolated RTX 3090 for precise measurements compared to four shared Tesla A100 for pre-training.

3.1. (In)Efficiency Analysis

The overall pre-training efficiency of the eight AFEs is evaluated for the first time on a carefully crafted benchmark to enable an extensive comparison of training costs.

Metrics and evaluation. Each AFE follows a pre-training for a single epoch over the two above-described datasets. A warm-up of 5 backward steps is used to avoid the *cudnn* optimization overhead. The mini-batch size is fixed to fit the memory budget. AFEs are compared over standard efficiency metrics including the averaged training time necessary to process the 1,000 sentences, the average forward and backward propagation times, the peak VRAM consumption, and the time necessary to process one second of speech including forward and backward computations. The latter metric expresses the throughput of the neural network. Measurements are obtained with full-precision operations on a single RTX 3090 24GB. Note however that our SSL AFEs can exhibit even smaller GPU requirements via low precision arithmetics, e.g., half-precision, or lower precision optimizers e.g. 8-bit [31] among others.

(In)Efficiency Results. The height of the bars in the upper parts of Figure 1 depicts the training time required with every AFE. Compared to wav2vec 2.0, all AFEs except Leaf offer significant speed-up in training time. In particular, discarding neural networks in the AFE part results in drastic forward speed improvements, as demonstrated by the 1 ms forward speed of the FBank AFE against 8 ms for wav2vec 2.0. As expected, only the AFE and backward pass are speeding up in most cases as the Transformer block remains unaltered. The FBank-CNN2d AFE, also changes this behavior as its output frequency is halved, resulting in a 2× speed-up for the Transformer forward pass as well. Leaf, unfortunately, suffers exhibits an increase in

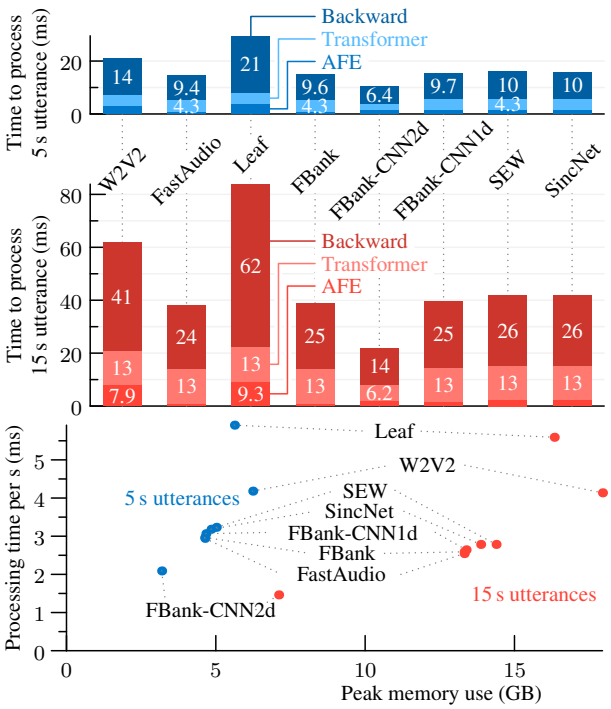

Figure 1: *Efficiency analysis of the eight AFEs. Time measurements are averages for processing 1,000 sentences. The stacked bar charts report forward and backward timings while the scatter plot offers a view of the throughput (i.e. averaged processing time in ms to train on one second of speech) against peak VRAM. Aside from Leaf, all proposed AFEs offer a training speed-up and a better throughput/VRAM ratio.*

processing time for all its components compared to the baseline. This is due to the lack of *cudnn* support in its implementation. Hence may expect a roughly halved training time in real pre-training conditions for the candidate AFE, excluding Leaf, compared to the baseline. Memory- and throughput-wise, FBank-CNN2d clearly outperforms the other AFEs by only consuming 7 GB of memory against 18 GB and 13 GB for wav2vec 2.0 and Fbank-CNN1d respectively on 15 seconds long utterances. The latter behavior is expected due to the lower time resolution of the model. This lowered time resolution also enables the Fbank-CNN2d to exhibit the highest gain in throughput when dealing with longer sentences as the processing time per second of speech decreases from 2.2 ms to 1.4 ms for 5 s and 15 s sentences respectively. Again, all AFE models except Leaf offer significant gains both in terms of VRAM consumption and throughput compared to the base wav2vec 2.0. Mini-batch size variations have also been investigated, yet not reported for the sake of readability. In fact, the less memory-demanding AFEs, e.g. FBank-CNN2d, enable much larger mini-batch sizes than the standard wav2vec 2.0 (e.g. 64 against 16 with 15 s long sentences). This results in better utilization of the GPU, smaller training times, and critically, enabling the training of speech SSL models on cheap GPU such as the Ti 80 and 90 families.

3.2. Downstream Evaluation

The downstream performances of seven AFEs are evaluated on three downstream tasks originating from the SUPERB benchmark. We decided to discard Leaf due to poor processing speed performance resulting in untractable training times.

Table 1: *Downstream results wav2vec 2.0 pre-training with seven AFEs (Leaf is discarded due to poor efficiency measurements). For speech recognition, word error rates (WER) after fine-tuning are on the subsets of Librispeech (*dev-clean *and* test-clean*). For keyword spotting, accuracies are on Google Speech Commands. For speaker verification, the equal error rate (EER) is after fine-tuning on VoxCeleb1. GPU requirements are for 200k steps with 4 GPUs in less than 7 days of pre-training, within GPU memory.*

Model	AFE output frequency	Speech recognition with LM Fine-tuning Time	*dev-clean* WER ↓	*test-clean* WER ↓	Keyword spotting Accuracy ↑	Speaker verification EER ↓	Requirements for pre-training GPU-hours ↓	Speed-up ↑	Minimum GPU number/type
wav2vec 2.0	49 Hz	6 h	6.0	6.5	98.2	4.1	670 h	1.0×	4 A100 80GB
SincNet	50 Hz	5 h	6.1	6.5	98.3	4.2	468 h	1.4×	4 V100 32GB
SEW	49 Hz	4 h	6.5	7.1	97.9	4.1	448 h	1.5×	4 V100 32GB
FastAudio	50 Hz	3 h	7.0	7.9	97.9	4.4	313 h	2.1×	4 V100 32GB
FBank-CNN1d	50 Hz	3 h	6.2	6.6	98.2	4.2	264 h	2.5×	4 V100 32GB
FBank	50 Hz	3 h	7.1	7.9	98.1	4.5	250 h	2.7×	4 V100 32GB
FBank-CNN2d	25 Hz	2 h	7.0	7.8	98.3	4.1	180 h	3.7×	4 3090 24GB

SSL pre-training. All models are pre-trained for 200k steps on the full Librispeech dataset (960 hours of speech). Indeed, and according to previous work on SSL efficiency [18], a full 400k pre-training may not be necessary to appreciate the downstream performance of different wav2vec 2.0 models. A full pre-training of the base wav2vec 2.0 requires four Nvidia Tesla A100 80GB for more than two weeks. Cutting the pre-training to 200k enables a fair comparison while limiting the maximal training time to slightly over a week, and reducing overall energy cost. The SSL pre-training follows the experimental protocol first described in [21]. Hence, the number of negatives (100), the length of the masks (10), their probability (0.65), and the 1.6 hours mini-batch length are identical.

Downstream evaluation. ASR training is performed with the 100 hours clean subset of Librispeech. Word error rates are reported on the dev and test clean sets with 4-gram rescoring. Two dense layers alongside layer normalization are added on top of the pre-trained model following the official SpeechBrain recipe. The entire architecture including wav2vec 2.0 is fine-tuned with the CTC loss. KS is done with the Google Speech Command dataset with the 12 actions set. Average pooling is applied to the output of the wav2vec 2.0 with a final dense layer for classification. Again, and in contrast to SUPERB, the whole architecture is fine-tuned from end-to-end. Finally, ASV is conducted with the VoxCeleb1 dataset. In this case, the wav2vec 2.0 is frozen to add some diversity to our evaluation. Hence, we follow the SUPERB approach and instead extract a learned weighted sum from all the layers of the wav2vec 2.0. Nevertheless, this evaluation uses the SOTA ECAPA-TDNN architecture [32] to consume those features instead of the mere x-vector [33] model from SUPERB. Hyper-parameters are identical to existing SpeechBrain recipes for each dataset.

Downstream results. First, the reported results in Table 1 are in-line with previous works evaluating partially pre-trained wav2vec 2.0 [18]. Speaker verification EER, for instance, are even better than those reported within the official SUPERB leaderboard with 6.0% against 4.1% in our experiments for the base wav2vec 2.0. The latter difference is due to the use of the ECAPA-TDNN architecture as a downstream decoder. Then, pre-training and Librispeech fine-tuning times also align with findings from the efficiency analysis with every AFE offering speed-ups ranging from $1.4\times$ for SincNet to $3.7\times$ for FBANK-CNN2d compared to wav2vec 2.0. Hence, the total pre-training time is lowered to 180 against 670 GPU hours for FBANK-CNN2d and wav2vec 2.0 respectively. Most newly introduced AFEs also lower the bar in terms of GPU requirements as they enable a 7 days or less pre-training on four Tesla V100 32GB or RTX 3090 24GB for the FBANK-CNN2d compared to four Tesla A100 80GB for the wav2vec 2.0. The latter change induces a decrease in the unitary GPU price of a factor of roughly $10\times$. Wav2vec 2.0, however, offers the most consistent downstream performance across the three tasks. We hypothesize that such behavior is the result of the extensive hyperparameter tuning at the origin of the wav2vec 2.0 architecture. Nevertheless, the drop in accuracies and word error rates observed with more efficient AFEs are far from dramatic. Indeed, FBANK-CNN1d obtains similar performance to the baseline with a $0.1\,\%$ relative increase in WER and EER and an equivalent accuracy on KS, while the pre-training time is reduced by $2.5\times$. The fastest and cheapest alternative, FBANK-CNN2d, suffers from higher WER with a relative increase of $1.3\,\%$, but also similar or identical ASV and KS performance compared to wav2vec 2.0. SEW, the standard FBANK, and FastAudio also offer important speed and memory improvements, but also show important increases in WER. ASV and KS accuracies also illustrate that the latter AFEs are totally suitable for these tasks. SincNet sits in the middle, as it offers the smallest speed improvement (i.e. $1.4\times$) but also exhibits comparable WER, EER and KS accuracy against wav2vec 2.0. Overall, Table 1 shows that certain AFE, such as FBANK-CNN1d or SincNet can significantly lower the pre-training cost of wav2vec 2.0-based SSL models while ensuring a similar level of downstream performance. There exist use cases where a relative gain of $0.1\,\%$ in raw performance with a $3\times$ increase in computing cost might not be justified. Hence, such results should encourage the community to integrate better descriptors of the model efficiency in standardized benchmarks.

4. Conclusion

Acoustic features extractors play an important role in the large compute and memory costs of speech SSL pre-training. This paper bridges the gap between the existing abundant speech processing literature and SSL pre-training and compares extensively three novel and five existing AFE, both at the efficiency and downstream performance levels. The observed results suggest that significant compute and memory savings can be achieved without disproportionated downstream performance impact. The latter finding is a major step forward to lowering the entry bar to speech SSL pre-training.

5. References

[1] L. Jing and Y. Tian, "Self-supervised visual feature learning with deep neural networks: A survey," *IEEE transactions on pattern analysis and machine intelligence*, vol. 43, no. 11, pp. 4037–4058, 2020.

[2] S. Sinha, A. Mandlekar, and A. Garg, "S4rl: Surprisingly simple self-supervision for offline reinforcement learning in robotics," in *Conference on Robot Learning*. PMLR, 2022, pp. 907–917.

[3] A. Mohamed, H.-y. Lee, L. Borgholt, J. D. Havtorn, J. Edin, C. Igel, K. Kirchhoff, S.-W. Li, K. Livescu, L. Maaløe *et al.*, "Self-supervised speech representation learning: A review," *IEEE Journal of Selected Topics in Signal Processing*, 2022.

[4] X. Liu, F. Zhang, Z. Hou, L. Mian, Z. Wang, J. Zhang, and J. Tang, "Self-supervised learning: Generative or contrastive," *IEEE Transactions on Knowledge and Data Engineering*, 2021.

[5] W.-N. Hsu, B. Bolte, Y.-H. H. Tsai, K. Lakhotia, R. Salakhutdinov, and A. Mohamed, "Hubert: Self-supervised speech representation learning by masked prediction of hidden units," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 29, pp. 3451–3460, 2021.

[6] S. Chen, C. Wang, Z. Chen, Y. Wu, S. Liu, Z. Chen, J. Li, N. Kanda, T. Yoshioka, X. Xiao *et al.*, "Wavlm: Large-scale self-supervised pre-training for full stack speech processing," *arXiv preprint arXiv:2110.13900*, 2021.

[7] S. Evain, H. Nguyen, H. Le, M. Z. Boito, S. Mdhaffar, S. Alisamir, Z. Tong, N. Tomashenko, M. Dinarelli, T. Parcollet, A. Allauzen, Y. Estève, B. Lecouteux, F. Portet, S. Rossato, F. Ringeval, D. Schwab, and laurent besacier, "Task agnostic and task specific self-supervised learning from speech with lebenchmark," in *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 2)*, 2021. [Online]. Available: https://openreview.net/forum?id=TSvj5dmuSd

[8] S. wen Yang, P.-H. Chi, Y.-S. Chuang, C.-I. J. Lai, K. Lakhotia, Y. Y. Lin, A. T. Liu, J. Shi, X. Chang, G.-T. Lin, T.-H. Huang, W.-C. Tseng, K. tik Lee, D.-R. Liu, Z. Huang, S. Dong, S.-W. Li, S. Watanabe, A. Mohamed, and H. yi Lee, "SUPERB: Speech Processing Universal PERformance Benchmark," in *Proc. Interspeech 2021*, 2021, pp. 1194–1198.

[9] Z. Chen, S. Chen, Y. Wu, Y. Qian, C. Wang, S. Liu, Y. Qian, and M. Zeng, "Large-scale self-supervised speech representation learning for automatic speaker verification," in *ICASSP 2022-2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2022, pp. 6147–6151.

[10] H. Nguyen, F. Bougares, N. Tomashenko, Y. Estève, and L. Besacier, "Investigating self-supervised pre-training for end-to-end speech translation," in *Interspeech 2020*, 2020.

[11] A. Babu, C. Wang, A. Tjandra, K. Lakhotia, Q. Xu, N. Goyal, K. Singh, P. von Platen, Y. Saraf, J. Pino, A. Baevski, A. Conneau, and M. Auli, "XLS-R: Self-supervised Cross-lingual Speech Representation Learning at Scale," in *Proc. Interspeech 2022*, 2022, pp. 2278–2282.

[12] G. Laperriere, V. Pelloin, M. Rouvier, T. Stafylakis, and Y. Estève, "On the use of semantically-aligned speech representations for spoken language understanding," *2022 IEEE Spoken Language Technology Workshop (SLT)*, pp. 361–368, 2022.

[13] Z. Huang, S. Watanabe, S.-w. Yang, P. García, and S. Khudanpur, "Investigating self-supervised learning for speech enhancement and separation," in *ICASSP 2022-2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2022, pp. 6837–6841.

[14] Y.-C. Wang, S. Venkataramani, and P. Smaragdis, "Self-supervised learning for speech enhancement," *arXiv preprint arXiv:2006.10388*, 2020.

[15] H.-S. Tsai, H.-J. Chang, W.-C. Huang, Z. Huang, K. Lakhotia, S.-w. Yang, S. Dong, A. Liu, C.-I. Lai, J. Shi *et al.*, "Superb-sg: Enhanced speech processing universal performance benchmark for semantic and generative capabilities," in *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 2022, pp. 8479–8492.

[16] Y. Gao, J. Fernandez-Marques, T. Parcollet, A. Mehrotra, and N. Lane, "Federated Self-supervised Speech Representations: Are We There Yet?" in *Proc. Interspeech 2022*, 2022, pp. 3809–3813.

[17] A. Baevski, W.-N. Hsu, Q. Xu, A. Babu, J. Gu, and M. Auli, "Data2vec: A general framework for self-supervised learning in speech, vision and language," in *International Conference on Machine Learning*. PMLR, 2022, pp. 1298–1312.

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

[19] T.-Q. Lin, H.-y. Lee, and H. Tang, "Melhubert: A simplified hubert on mel spectrogram," *arXiv:2211.09944*, 2022.

[20] 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 *ICASSP 2022 - 2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2022, pp. 7667–7671.

[21] A. Baevski, Y. Zhou, A. Mohamed, and M. Auli, "wav2vec 2.0: A framework for self-supervised learning of speech representations," *Advances in Neural Information Processing Systems*, vol. 33, pp. 12 449–12 460, 2020.

[22] M. Ravanelli, T. Parcollet, P. Plantinga, A. Rouhe, S. Cornell, L. Lugosch, C. Subakan, N. Dawalatabad, A. Heba, J. Zhong *et al.*, "Speechbrain: A general-purpose speech toolkit," *arXiv preprint arXiv:2106.04624*, 2021.

[23] Y. Gao, J. Fernandez-Marques, T. Parcollet, P. P. de Gusmao, and N. D. Lane, "Match to win: Analysing sequences lengths for efficient self-supervised learning in speech and audio," *IEEE Spoken Language Technology Workshop*, 2022.

[24] A. Lawson, P. Vabishchevich, M. Huggins, P. Ardis, B. Battles, and A. Stauffer, "Survey and evaluation of acoustic features for speaker recognition," in *2011 IEEE international conference on acoustics, speech and signal processing (ICASSP)*. IEEE, 2011, pp. 5444–5447.

[25] O. Abdel-Hamid, A.-r. Mohamed, H. Jiang, and G. Penn, "Applying convolutional neural networks concepts to hybrid nn-hmm model for speech recognition," in *2012 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2012, pp. 4277–4280.

[26] S. K. Kopparapu and M. Laxminarayana, "Choice of mel filter bank in computing mfcc of a resampled speech," in *10th International Conference on Information Science, Signal Processing and their Applications (ISSPA 2010)*. IEEE, 2010, pp. 121–124.

[27] A. Radford, J. W. Kim, T. Xu, G. Brockman, C. McLeavey, and I. Sutskever, "Robust speech recognition via large-scale weak supervision," *arXiv preprint arXiv:2212.04356*, 2022.

[28] Q. Fu, Z. Teng, J. White, M. E. Powell, and D. C. Schmidt, "Fastaudio: A learnable audio front-end for spoof speech detection," in *ICASSP 2022-2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2022, pp. 3693–3697.

[29] N. Zeghidour, O. Teboul, F. de Chaumont Quitry, and M. Tagliasacchi, "Leaf: A learnable frontend for audio classification," in *International Conference on Learning Representations*.

[30] T. Parcollet, M. Morchid, and G. Linares, "E2e-sincnet: Toward fully end-to-end speech recognition," in *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2020, pp. 7714–7718.

[31] T. Dettmers, M. Lewis, S. Shleifer, and L. Zettlemoyer, "8-bit optimizers via block-wise quantization," in *International Conference on Learning Representations*.

[32] B. Desplanques, J. Thienpondt, and K. Demuynck, "Ecapa-tdnn: Emphasized channel attention, propagation and aggregation in tdnn based speaker verification," *Proc. Interspeech 2020*, pp. 3830–3834, 2020.

[33] D. Snyder, D. Garcia-Romero, G. Sell, D. Povey, and S. Khudanpur, "X-vectors: Robust dnn embeddings for speaker recognition," in *2018 IEEE international conference on acoustics, speech and signal processing (ICASSP)*. IEEE, 2018, pp. 5329–5333.