Interspeech 2022
18-22 September 2022, Incheon, Korea

Investigation of Ensemble features of Self-Supervised Pretrained Models for Automatic Speech Recognition

A Arunkumar, Vrunda N Sukhadia*, S. Umesh*

Speech Lab, Dept. of Electrical Engineering, IIT Madras, Chennai, India
arunkumaras10@gmail.com, sukhadiavrunda@gmail.com, umeshs@ee.iitm.ac.in

Abstract

Self-supervised learning (SSL) based models have been shown to generate powerful representations that can be used to improve the performance of downstream speech tasks. Several state-of-the-art SSL models are available, and each of these models optimizes a different loss which gives rise to the possibility of their features being complementary. This paper proposes using an ensemble of such SSL representations and models, which exploits the complementary nature of the features extracted by the various pretrained models. We hypothesize that this results in a richer feature representation and show results for the ASR downstream task. To this end, we use three SSL models that have shown excellent results on ASR tasks, namely HuBERT, Wav2Vec2.0, and WavLM. We explore the ensemble of models fine-tuned for the ASR task and the ensemble of features using the embeddings obtained from the pre-trained models for a downstream ASR task. We get a relative improvement of 10% in ASR performance over individual models and pre-trained features when using *LibriSpeech*(100h) and WSJ dataset for the downstream tasks.

Index Terms: self supervised learning, ensemble features, feature extraction,

1. Introduction

Self-Supervised Learning (SSL) models [1–10] have been shown to provide significant improvement for various downstream tasks such as Automatic Speech Recognition(ASR), phoneme recognition, speaker verification, etc. Progress in ASR in many languages has been significantly affected due to the lack of good quality transcribed data. Self-supervised methods are highly desirable in such scenarios since they need a large amount of audio-only data, which is more readily available. Several pre-trained SSL models are now available in the public domain that can be used to build good downstream ASR models. These models vary in the loss function used during self-supervised learning and the type, nature, and amount of data used in self-supervision. We look at three models that have state-of-art ASR results for *LibriSpeech* subsets as well on SUPERB benchmarks [6, 9–11], namely Wav2Vec2.0, HuBERT, and WavLM. Each of these self-supervised training methods is motivated by a different objective:

- Wave2vec2.0 masks latent representation and solves a contrastive task with respect to the quantized version of the latent representation
- HuBERT discovers acoustic units using a clustering approach to construct target labels corresponding to the input features. Masking is then applied to the input features, and the training is done to minimize the masked prediction loss using cluster labels as targets
- WavLM adds gated relative position bias to the transformer structure, and apart from masked prediction loss similar to HuBERT, also applies denoising task during self-supervised learning.

The last two models use non-contrastive criteria and therefore do not require large batch sizes, which is essential during the training of Wav2Vec2.0 models. As different SSL models optimize different objective functions, each of them learns to extract different set of features. This opens the possibility of the extracted features being complementary. This paper investigates an ensemble method that combines such complementary features from different pre-trained self-supervised models. Apart from the different objective functions, each of these models has been trained on varying amounts of unlabelled data that are readily available in the public domain. Similarly, many of these models are also available after fine-tuning for various ASR tasks.

The following state-of-the-art self-supervised models, which are readily available to download from HuggingFace [12] are used in this paper:

- "facebook/wav2vec2-base" pre-trained on *LibriSpeech* 960hrs unlabeled data [95M parameters]
- "facebook/hubert-base-ls960" pre-trained on *LibriSpeech* 960hrs unlabeled data [95M parameters]
- "microsoft/wavlm-base" pre-trained on *LibriSpeech* 960hrs unlabeled data [94.70 parameters]
- "facebook/wav2vec2-large-lv60" pre-trained on *Libri-Light* 60k hours unlabeled data [317.3M parameters]
- "facebook/hubert-xlarge-ll60k" pre-trained on *Libri-Light* 60k hours unlabeled data [316.6M parameters]
- "microsoft/wavlm-large" pre-trained on mix 94k hours unlabeled data [60k hrs *Libri-Light* + 10k hrs *GigaSpeech* + 24k hrs *VoxPopuli*] [316.6M parameters]
- Finetuned models "facebook/wav2vec2-large-960h-lv60-self" and "facebook/hubert-xlarge-ls960-ft" which are pre-trained on *Libri-Light* 60k hours unlabeld data and finetuneed on *LibriSpeech* 960 hours labeld data.

Librispeech-100-clean [13] and *WSJ* are used for downstream tasks.

2. Fine Tuning for ASR task

Pre-trained self-supervised models can be used in downstream ASR tasks in two ways:

- Finetuning the pre-trained self-supervised model using the supervised downstream dataset. This is done by

*These authors have contributed equally.

10.21437/Interspeech.2022-11376

adding a linear CTC layer [14] on top of the embeddings of the last layer output of the pre-trained model and optimizing for the character output obtained from supervision. In this approach all the parameters of the pre-trained model are also updated (after freezing for a few iterations), and is usually expensive. For example, Wav2Vec2.0 finetuning on *LibriSpeech* splits take at least 50 V100 GPU hours [6]

- Freezing the pre-trained self-supervised model and use extracted features for downstream tasks. Since the SSL model parameters are not updated, a simple linear CTC layer on top during finetuning is insufficient to get good results. In practice, a couple of BiLSTM layers or transformer encoder layers are added on top of the pre-trained features, followed by a CTC layer. During finetuning, the BiLSTM/Transformer layers and the CTC layers are optimized for the supervised character output. A learned linear combination of multiple SSL layer outputs is often used as features before feeding to the transformer encoder layers.

This paper considers the strategy of freezing the pre-trained self-supervised model and using the models or extracted features for ASR tasks. Ensemble learning is a common approach in Deep Learning since it is predicated on the premise that combining the output of numerous models is more effective than using a single model and typically produces superior results. There are various approaches to combining features from multiple models, including 1) Summation of Features, 2) Weighted Average of Features, 3) Concatenation of Features, and 4) Soft mixing using an Attention layer. In this paper, we combine features from different models by concatenating extracted features and using the linear layer with CTC loss to learn the optimal combination of these concatenated features. However, since the SSL models are frozen, using only the CTC loss is not enough to guide the best feature vector selection. Therefore, we also propose to use a Transformer encoder [15] on top of the ensemble of features to compute a soft mixture of the features; this allows for an even richer representation of features. A similar strategy is also adopted in S3PRL and SUPERB benchmarks. We first describe an approach to combining SSL models that have been finetuned for the ASR task. This is followed by a method to combine embeddings obtained from different pre-trained (but not finetuned) SSL models for a downstream ASR task.

2.1. Ensemble Model

In this section, we describe our approach to combining SSL models that have been finetuned for the ASR task. Specifically, we consider the Wav2Vec2.0 and HuBERT model finetuned on the *LibriSpeech*-960 hour supervised data. These models have been finetuned with a CTC Linear layer on top of the pre-trained transformer encoders, with all the SSL model parameters being updated during finetuning (frozen for some initial iterations). In our proposed approach, we remove the final CTC layers from both the finetuned models and concatenate the two final layer embeddings. We now add a randomly initialized CTC linear layer on top of these concatenated features and finetune on a small amount of training data for a few epochs. Since these are already finetuned models, few epochs of finetuning help learn the mapping from concatenated features to characters. Please note that the finetuned SSL model parameters are *not* updated, and only the CTC layer parameters on top of the concatenated features are learned. This is shown in Figure 1.

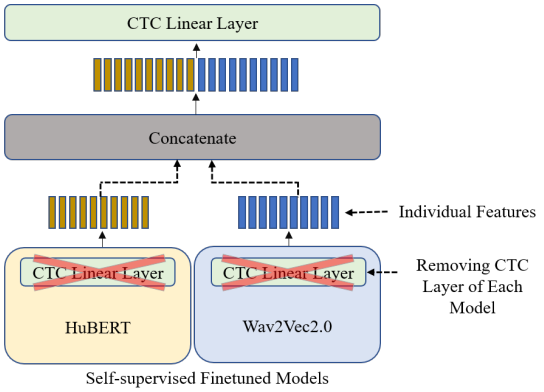

Figure 1: *Proposed method for Ensemble model to combine HuBERT and Wav2Vec2.0 fine-tuned ASR models.*

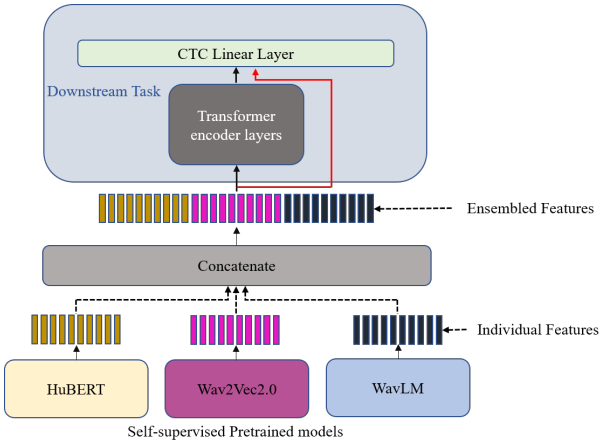

Figure 2: *Proposed method to combine the embeddings from pre-trained models for downstream ASR task*

2.2. Ensemble Features

This section describes the method to combine the features or embeddings from different pre-trained models for a downstream ASR task. Note that unlike previous section, these are pre-trained models only, and *not* fine-tuned on any ASR task. Again, we freeze the pre-trained model parameters and do *not* allow them to update. Therefore, a simple CTC layer on top of embeddings during finetuning will not give good performance. We need a few learnable layers on top to achieve good performance. This is similar to the approach taken by S3PRL and SUPERB benchmarks [11], where a bi-directional LSTM on top of a frozen pre-trained model is used for finetuning ASR downstream task. The authors proposed to freeze the pre-trained model and use the weighted average of all the hidden layer features with trainable weights. Instead of combining features from different layers of the same model, this work proposes to combine the last layer features of different pre-trained models and pass these features to transformer encoder layers. We use transformer encoders instead of BiLSTM since it offers faster processing with improved representation. The proposed ensemble is generated by combining embeddings from multiple state-of-the-art models. All models in the ensemble are frozen, and feature representation collected from the final layer of each model is concatenated to produce an ensemble. This is shown

in Figure 2.

3. Experiments

3.1. Ensemble Model

For all the experiments presented in this section, large variants of wav2vec2 and HuBERT pre-trained with Libri-Light 60k hours and finetuned with *LibriSpeech* 960h are considered. These finetuned models are available for download in HuggingFace [12]. All our experiments are done on the SpeechBrain toolkit [16]. The CTC layer is removed from these models, and the last layer features are extracted from these models. Note that these features are obtained from well finetuned ASR models and therefore carry rich information about the final classification task of characters. While Wav2Vec2.0 has 1024 dimension embeddings, HuBERT has 1280 dimension embeddings. In our proposed approach, these features are concatenated before feeding them to a new randomly initialized CTC layer. The parameters of the linear classification layer are learned using a few epochs on a small amount of training data. The linear layer helps learn an appropriate combination of the concatenated features and a mapping of these features to characters. To enable a fair comparison, the features from the individual models are obtained by removing the CTC layer of the finetuned model. These features are fed to a randomly initialized CTC layer whose parameters are learned by running a few epochs on a small amount of training data.

Table 1 shows the result of combining the finetuned models using the approach of Figure 1. In this case, we have finetuned the randomly initialized CTC layer using 10 hours of *LibriSpeech* data for about 8 epochs. The remaining model parameters are frozen. As seen from the table, using the ensemble of models gives significant improvement over using individual models. Although both our models have been finetuned on 960-hour *LibriSpeech*, we also test them on WSJ data using the same approach. Once again, we find improved performance using the proposed approach compared to the individual models.

Method	**LibriSpeech**				**WSJ**	
	clean		**other**		**test -Dev 93**	**test -Eval 93**
	dev	**test**	**dev**	**test**		
Wav2Vec 2.0	4.13	4.31	6.91	7.21	9.39	8.83
HuBERT	4.35	4.03	5.79	6.30	7.32	6.54
Wav2Vec 2.0 + HuBERT	3.15	3.05	4.76	5.16	6.79	6.13

Table 1: *Evaluating Results Of Individual SSL Methods And Ensemble Model For In-domain (Libri-Light 10h) And Out-domain (10h Subset Of WSJ) Downstream Data. Only CTC Layer Is Added On Top Of Features. No External Language Model is Used.*

3.2. Ensemble Features

In this section, we present the results of using the ensemble of features from pre-trained models to improve the performance of the downstream ASR task. Both the base and large variants of pre-trained models are used for the ensemble feature experiments. In this section, apart from Wav2Vec2.0 and HuBERT, we also consider the pre-trained WavLM model since this is also readily available in HuggingFace [12]. All these experiments were implemented in speechbrain toolkit [16]. The last layer features are extracted from the pre-trained model and then passed to the CTC layer or transformer network. Note that all the pre-trained models we are using have been trained on *LibriSpeech* data in a self-supervised manner. We now conduct experiments in two scenarios: (i) where the fine-tuning is done on matched data, namely *LibriSpeech* subsets, and (ii) where the fine-tuning is done a different domain data, namely WSJ. Since these embeddings are obtained from pre-trained models that have not been tuned for ASR task, and since we do not allow the pre-trained model parameters to be updated, a simple CTC linear on top does not give good performance. We need additional learnable layers to get good performance, and in our case, we use transformer encoder layers.

3.2.1. Finetuning with Same Domain Data

In this section, the pre-trained models are fine-tuned with data from same domain, namely, *LibriSpeech*-100 hour data.

Feature Extraction Model	**CTC Layer only**				**2 Encoder Layers + CTC**			
	Clean		**Other**		**Clean**		**Other**	
	dev	**test**	**dev**	**test**	**dev**	**test**	**dev**	test
Base models								
WavLM	59.54	60.19	67.73	67.85	9.65	10.34	20.87	20.88
HuBERT	67.96	68.43	76.33	76.2	12.41	13.37	24.8	25.16
WavLM +HuBERT	51.02	51.57	60.43	60.59	8.72	9.25	19.63	19.60
WavLM +Wav2Vec2.0	60.08	60.64	68.43	68.57	11.58	12.59	23.38	23.60
Wav2Vec2.0 +HuBERT	68.59	68.49	76.95	76.19	13.77	14.34	26.19	27.0
Wav2Vec2.0 +HuBERT +WavLM	52.21	52.87	61.53	61.73	9.03	9.58	19.96	20.16
Large models								
WavLM	47.02	47.66	54.04	53.78	5.79	5.97	11.09	11.01
HuBERT	58.98	58.44	62.64	62.19	9.94	10.36	13.54	13.93
WavLM +HuBERT	41.17	41.61	46.38	46.06	5.60	5.49	9.76	9.53
WavLM +Wav2Vec2.0	47.44	48.01	53.94	53.89	6.64	6.99	12.22	12.21
Wav2Vec2.0 +HuBERT	56.47	55.84	66.77	60.69	9.64	9.90	12.96	13.35
Wav2Vec2.0 +HuBERT +WavLM	40.00	40.24	46.21	46.37	5.60	5.66	9.76	9.83

Table 2: *Ablation study for Ensemble Features with 1) CTC linear Layer 2) Two Encoder Layers + CTC Linear Layer for LibriSpeech Dataset*

A CTC-only model or a downstream transformer encoder model with CTC is finetuned with the individual and ensemble features extracted from *LibriSpeech* 100h data. In the first set of experiments, only a CTC layer is finetuned. In the second set of experiments, two-layer transformer encoder with a CTC layer on top is finetuned. The results are shown in Table 2. Since the features are extracted from a pre-trained model and not from a finetuned model, CTC only finetuning is not good enough. It can be seen that the ensemble features model provides improvement over the individual features model. Even though the

performance is poor in the case of CTC only finetuning, the ensemble features method is still relatively better than the individual features. The more interesting results are with transformer encoder layers trained on top of these features. From Table 2, the following observations can be made:

- SSL Models trained on larger amounts of data consistently performed better than base models on the same domain downstream task as expected.
- WavLM consistently performs better than HuBERT, which is better than Wav2Vec2.0 in all cases for this ASR task.
- Wav2vec2.0 features are significantly worse, and therefore hurt performance when combined with other features. This is because, as observed in previous works, the final layer of WavLM and HuBERT capture significant features for ASR task, while it is the middle layers of Wav2Vec2.0 that are more appropriate for ASR task. In our experiments, since we have used final layer embeddings in all cases and models, Wav2vec2.0 provides degraded performance.
- Combining WavLM and HuBERT features gives the best performance for this task and is significantly better than the individual models.
- Compare to the best individual model performance, the WavLM+HuBERT features give a relative 10% improvement.

3.2.2. Finetuning with Different Domain Data

In this section, we finetune using WSJ data for the downstream ASR task. While WSJ data is mainly from the business domain with modern English, *LibriSpeech* is mostly audiobooks of old English material from project Gutenberg. Therefore, there is a mismatch in the domain. Since the domain of labeled data used for finetuning is different from the data used for pre-training, we need a more complex model to learn better. Therefore, while for the previous section, we got good performance with a 2-layer encoder, for the mismatched WSJ task, we used an 8-layer encoder followed by CTC to get good performance. Only a CTC layer is applied on top and finetuned in the first set of experiments. As expected, these perform far worse in this mismatched case. In the next set of experiments, an eight-layer transformer encoder is finetuned. From Table 3 we make the following observations:

- For fine-tuning models from a different domain too, the WavLM model performs better than HuBERT, which is better than Wav2Vec2.0
- In this case also, a combination of WavLM+HuBERT gives the best performance.
- The relative improvement of the ensemble features method is more significant while using large pre-trained models than while using base models. In the case of mismatched WSJ data, it is only the large models that provide gains when used in an ensemble.
- Even for the WSJ task, the ensemble features method provides about 10% relative improvement over the best performing individual features.

Feature Extraction Model	**CTC Layer only**		**8 Encoder Layers + CTC**	
	Test-Dev93	**Test-Eval93**	**Test-Dev93**	**Test-Eval93**
Base Models				
WavLM	66.81	65.28	15.77	14.93
HuBERT	74.97	75.01	22.54	20.69
WavLM +HuBERT	59.46	58.05	16.04	14.93
WavLM +Wav2Vec2.0	68.05	67.37	16.24	15.37
Wav2Vec2.0 +HuBERT	74.91	74.87	17.94	17.66
Wav2Vec2.0 +HuBERT +WavLM	60.14	59.79	16.4	15.43
Large Models				
WavLM	55.83	55.08	11.09	10.37
HuBERT	66.97	66.47	13.22	12.73
WavLM +HuBERT	49.81	49.30	9.57	8.86
WavLM +Wav2Vec2.0	56.80	56.13	10.32	9.59
Wav2Vec2.0 +HuBERT	66.68	65.43	18.50	17.75
Wav2Vec2.0 +HuBERT +WavLM	48.79	47.62	15.09	13.89

Table 3: *Ablation study for Ensemble Features with 1) CTC linear Layer 2) Eight Encoder Layers + CTC Linear Layer for WSJ Dataset*

4. Conclusion

In this paper, we have explored the use of an ensemble of models and embedding from pre-trained models to improve the performance of individual SSL methods. In all cases, we have used publicly available models. The motivation for using ensemble is that different SSL methods employ different objective functions such as masked prediction loss or contrastive loss. Therefore, they may capture complementary information. On the downstream ASR, the use of our proposed approaches provides a relative improvement of 10% over the best individual models for both *LibriSpeech*-100 and WSJ task.

5. Acknowledgement

We would like to thank Metilda N J for the technical discussion and her help in preparing this paper.

6. References

[1] A. v. d. Oord, Y. Li, and O. Vinyals, "Representation learning with contrastive predictive coding," *arXiv preprint arXiv:1807.03748*, 2018.

[2] Y.-A. Chung, W.-N. Hsu, H. Tang, and J. Glass, "An unsupervised autoregressive model for speech representation learning," *arXiv preprint arXiv:1904.03240*, 2019.

[3] S. Schneider, A. Baevski, R. Collobert, and M. Auli, "wav2vec: Unsupervised pre-training for speech recognition," *arXiv preprint arXiv:1904.05862*, 2019.

[4] A. Baevski, S. Schneider, and M. Auli, "vq-wav2vec: Self-supervised learning of discrete speech representations," *arXiv preprint arXiv:1910.05453*, 2019.

[5] A. T. Liu, S.-w. Yang, P.-H. Chi, P.-c. Hsu, and H.-y. Lee, "Mockingjay: Unsupervised speech representation learning with deep bidirectional transformer encoders," in *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2020, pp. 6419–6423.

[6] 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.

[7] M. Ravanelli, J. Zhong, S. Pascual, P. Swietojanski, J. Monteiro, J. Trmal, and Y. Bengio, "Multi-task self-supervised learning for robust speech recognition," in *ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2020, pp. 6989–6993.

[8] A. T. Liu, S.-W. Li, and H.-y. Lee, "Tera: Self-supervised learning of transformer encoder representation for speech," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 29, pp. 2351–2366, 2021.

[9] 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.

[10] 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.

[11] S.-w. 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 *et al.*, "Superb: Speech processing universal performance benchmark," *arXiv preprint arXiv:2105.01051*, 2021.

[12] T. Wolf, L. Debut, V. Sanh, J. Chaumond, C. Delangue, A. Moi, P. Cistac, T. Rault, R. Louf, M. Funtowicz, J. Davison, S. Shleifer, P. von Platen, C. Ma, Y. Jernite, J. Plu, C. Xu, T. Le Scao, S. Gugger, M. Drame, Q. Lhoest, and A. Rush, "Transformers: State-of-the-art natural language processing," in *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*. Online: Association for Computational Linguistics, Oct. 2020, pp. 38–45. [Online]. Available: https://aclanthology.org/2020.emnlp-demos.6

[13] V. Panayotov, G. Chen, D. Povey, and S. Khudanpur, "Librispeech: an asr corpus based on public domain audio books," in *2015 IEEE international conference on acoustics, speech and signal processing (ICASSP)*. IEEE, 2015, pp. 5206–5210.

[14] A. Graves, S. Fernández, F. Gomez, and J. Schmidhuber, "Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks," in *Proceedings of the 23rd international conference on Machine learning*, 2006, pp. 369–376.

[15] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," *Advances in neural information processing systems*, vol. 30, 2017.

[16] 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.