Interspeech 2022
18-22 September 2022, Incheon, Korea

FeaRLESS: Feature Refinement Loss for Ensembling Self-Supervised Learning Features in Robust End-to-end Speech Recognition

Szu-Jui Chen, Jiamin Xie, John H.L. Hansen

Center for Robust Speech Systems, University of Texas at Dallas, TX 75080
{SzuJui.Chen, Jiamin.Xie, John.Hansen}@utdallas.edu

Abstract

Self-supervised learning representations (SSLR) have resulted in robust features for downstream tasks in many fields. Recently, several SSLRs have shown promising results on automatic speech recognition (ASR) benchmark corpora. However, previous studies have only shown performance for solitary SSLRs as an input feature for ASR models. In this study, we propose to investigate the effectiveness of diverse SSLR combinations using various fusion methods within end-to-end (E2E) ASR models. In addition, we will show there are correlations between these extracted SSLRs. As such, we further propose a feature refinement loss for decorrelation to efficiently combine the set of input features. For evaluation, we show that the proposed "FeaRLESS learning features" perform better than systems without the proposed feature refinement loss for both the WSJ and Fearless Steps Challenge (FSC) corpora.

Index Terms: End-to-End Speech Recognition, Self-Supervised Learning Representation, Feature Decorrelation

1. Introduction

Deep neural networks (DNNs) have accelerated the progress in speech processing, thus enhancing the accessibility of speech-related technologies in daily life. Since DNNs are data hungry, researchers have been exploring ways to enlarge the model and use more transcribed data. However, collecting a sufficient amount of labeled data for training is generally not feasible. As a result, unsupervised and semi-supervised learning have been proposed for utilizing unlabeled data. Previous studies in computer vision (CV) and natural language processing (NLP) have shown promising results using unsupervised learning to learn representations from data [1, 2]. There is a similar trend in the speech field based on pre-trained models using unlabeled data for learning powerful speech features. Recently, a series of self-supervised representation learning models have been proposed. The extracted representations are usually referred to as self-supervised learning representation (SSLR), and have achieved state-of-the-art results on several benchmarks [3, 4, 5, 6].

In the study by Chang et al. [3], we see an advantage of using SSLRs over traditional hand-crafted features. With this observation, an obvious question arises: Would it be better to combine SSLRs for automatic speech recognition (ASR)? We explore an answer to this question by conducting experiments using alternate fusion methods for SSLRs. However, our preliminary results with the Wall Street Journal (WSJ) corpus showed that simple fusion methods do not improve word error rate (WER).

After further investigation, we suggest two possible reasons, where the combination of powerful SSLRs does not surpass the baseline. The first is that the WSJ corpus might be too simple for a fusion method to be beneficial. The second is that the extracted SSLRs have certain correlations with each other that lead to redundant information. In order to efficiently combine SSLRs, we propose a loss function to decorrelate features as a feature redundancy reduction method for end-to-end (E2E) ASR model. In addition, we further conduct the same experiments on the Fearless Steps Challenge (FSC) Phase 2 corpus which is spontaneous natural noisy speech.

The contribution of our work includes:

- Investigate SSLR fusion in E2E ASR models
- Propose a feature refinement loss for efficient feature combining

2. Background

2.1. Self-Supervised Learning Representations

In this study, we consider objective function to categorize SSLRs, the main two being reconstructive loss and contrastive loss. For simplicity, we only note the SSLRs that are used in our experiments here, which are Hubert, Wav2Vec2.0, APC, TERA, and CPC.

Reconstructive Models trained with reconstructive loss can either predict future frames conditioned on the past history, or they can predict current frames conditioned on the past and future information based on masking prediction concepts. SSLRs that belong to this category include autoregressive predictive coding (APC) [7] and TERA [8].

Contrastive Alternatively, models trained with contrastive loss aim to distinguish positive from negative samples. SSLRs belonging to this category include contrastive predictive coding (CPC) [9] and Wav2Vec2.0 [6].

Finally, an exceptional model named Hubert [5] was recently proposed that used masked language modeling and pseudo-labeling for speech representation learning. For more details of these SSLRs, please refer to [3].

2.2. Feature Combination

Earlier research studies have explored the effects of feature combination compared with model ensemble techniques in traditional hybrid systems [10]. Various feature combination methods have been proposed, including linear discriminant analysis (LDA) [11] and concatenation, but only hand-crafted features such as MFCC and FBANK were explored. Recently, several studies have considered combining SSLRs with hand-crafted features [12, 13]. To date however, none of these investigations have explored combinations of SSLRs. Hence, in this study we explore SSLR fusion experiments that utilize feature combination techniques and analyze their effectiveness for E2E speech recognition.

10.21437/Interspeech.2022-10917

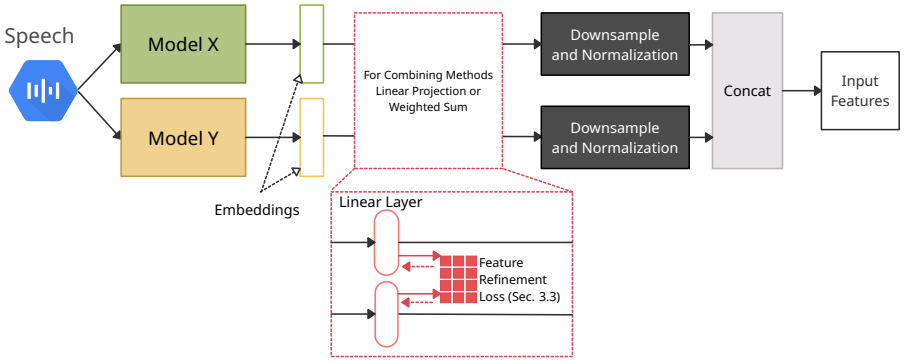


Figure 1: *Frontend feature extractor flow chart*

2.3. Decorrelation

In this last area, the idea of decorrelation is noted, where earlier efforts have used this for model ensemble [14] and preventive overfitting [15]. In the field of speech, past efforts explored feature decorrelation methods for speech recognition [16, 17, 18], but only consider single feature decorrelation. Inspired by these previous works, we explore advancements using decorrelation within a feature combination scenario for redundancy reduction. We propose a loss function for decorrelation between input features, in an effort to obtain a refined feature that is efficiently fused from SSLRs (details highlighted in Sec. 3.3).

3. Methods

In this section, we introduce several methods to combine SSLR features. Without loss of generality, we describe the case of combining only two features, $\boldsymbol{U} \in \mathbb{R}^{\mathrm{T}\times\mathrm{K}_1}$ and $\boldsymbol{V} \in \mathbb{R}^{\mathrm{T}\times\mathrm{K}_2}$, where T is the input feature length and K_1 and K_2 are the individual feature dimensions. The resolution of distinct features can be kept the same via downsampling. Since SSLR features can have different dimension sizes, we use an additional affine transformation to unify the feature dimension size to a common lower dimension. This allows the combined feature to be flexibly mapped to any feature subspace of a downstream model. In the following sections, we will categorize combination methods by whether or not one requires an affine transformation.

3.1. Combination Without Transformation

3.1.1. Concatenation

Concatenation is a natural approach for combining features without the need of a unified dimension size. The original information in each SSLR feature is retained by concatenation. Consider an output feature $\boldsymbol{X} \in \mathbb{R}^{\mathrm{T}\times(\mathrm{K}_1+\mathrm{K}_2)}$, our method can be written as:

$$\boldsymbol{X} = \left[norm(\boldsymbol{U}), norm(\boldsymbol{V})\right], \tag{1}$$

where $norm(\cdot)$ is the mean normalization along time T, using the statistics from each utterance. The reason to normalize before combination methods is to remove any bias learned within individual SSLRs. Removing bias allows features to be on the same scale and benefits the subsequent combining mechanisms, fusing them to the subspace of a downstream task. It is worth noting that many previous studies [19, 20] use feature concatenation extensively.

3.2. Combination With Transformation

For a combination of features with varying dimensions, we introduce learnable parameters for transforming features to a common dimension size. Specifically, we let the network learn two affine transformations with specific weights and biases: $\{\boldsymbol{W_1} \in \mathbb{R}^{\mathrm{K}_1\times\mathrm{K}}, b_1 \in \mathbb{R}^{1\times\mathrm{K}}, \boldsymbol{W_2} \in \mathbb{R}^{\mathrm{K}_2\times\mathrm{K}}, b_2 \in \mathbb{R}^{1\times\mathrm{K}}\}$. Using our previous notations, the transformed features are now $\tilde{\boldsymbol{U}} \in \mathbb{R}^{\mathrm{T}\times\mathrm{K}}$ and $\tilde{\boldsymbol{V}} \in \mathbb{R}^{\mathrm{T}\times\mathrm{K}}$, which are obtained as,

$$\tilde{\boldsymbol{U}} = \boldsymbol{U}\cdot\boldsymbol{W_1} + b_1, \tag{2}$$

$$\tilde{\boldsymbol{V}} = \boldsymbol{V}\cdot\boldsymbol{W_2} + b_2. \tag{3}$$

3.2.1. Linear Projection

Linear projection is the concatenation method with transformed features. Given an output feature $\boldsymbol{X} \in \mathbb{R}^{\mathrm{T}\times 2\mathrm{K}}$, our method is written as,

$$\boldsymbol{X} = \left[norm(\tilde{\boldsymbol{U}}), norm(\tilde{\boldsymbol{V}})\right], \tag{4}$$

where $norm(\cdot)$ is the mean normalization along time T. Comparing with the output feature in Eq. 1, the feature transformation enables further control of the input features, such as for feature selection and refinement, which are described in Sec. 3.3.

3.2.2. Weighted Sum

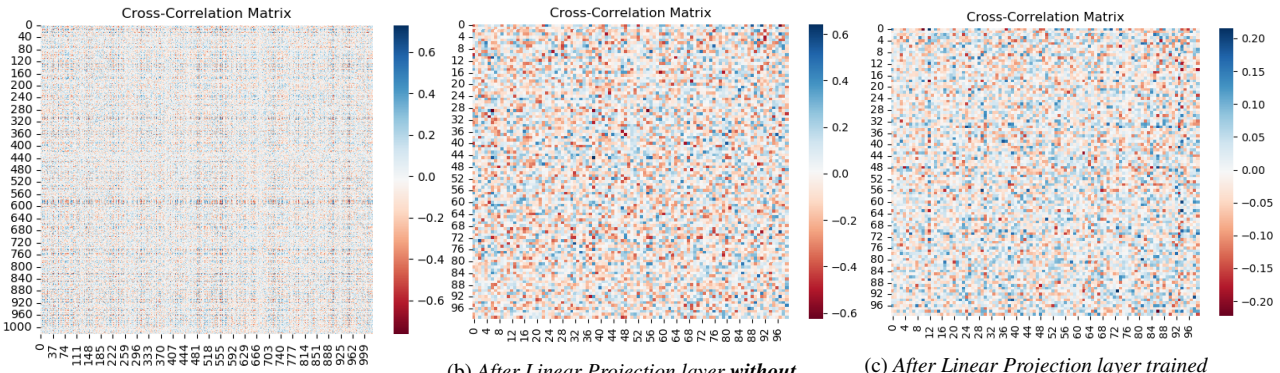
In the weighted sum approach, transformed features are first scaled and then combined. The importance among input features is controlled by two learnable scalars α and β. For an output feature $\boldsymbol{X} \in \mathbb{R}^{\mathrm{T}\times\mathrm{K}}$, our method can be written as,

$$\boldsymbol{X} = \frac{1}{\alpha+\beta}\cdot\left[norm(\tilde{\boldsymbol{U}}) \quad norm(\tilde{\boldsymbol{V}})\right]\cdot\begin{bmatrix}\alpha\\ \beta\end{bmatrix}, \tag{5}$$

where $norm(\cdot)$ is the mean normalization along time T. In comparison to linear projection in Eq. 4, the weighted sum adds more parameters to control the combination, where prior knowledge can be applied if necessary.

3.3. Feature Refinement Loss

The feature refinement loss guides the learning of feature transformations. As an ensemble strategy, the feature combination can benefit from a diverse set of input representations. However, we discover that non-trivial correlations can exist between SSLR features. For a random utterance, the correlation between extracted features from Hubert and Wav2Vec2.0 can reach over



(a) *Before Linear Projection*

(b) *After Linear Projection layer* ***without*** *feature refinement loss*

(c) *After Linear Projection layer trained* ***with*** *feature refinement loss*

Figure 2: *Cross-correlation matrix between Hubert and Wav2Vec2.0 of a randomly drawn utterance with the id of 4kac030m in WSJ dev03 set. The y-axis shows Hubert while the x-axis is Wav2Vec2.0. Sub-figure (a) uses raw features extracted from the pre-trained model. Sub-figure (b) and (c) compare the outcome of using a feature refinement loss, where (c) employs a threshold* $\epsilon = 0.2$*.*

0.6, considered as a high correlation, as shown in Fig. 2a. It becomes even clearer in the lower dimensions after the linear projection layer, as shown in Fig. 2b. Since highly correlated features represent similar information, they are generally considered redundant for a system and especially for an ensemble. By introducing the feature refinement loss, we manage to reduce the correlation as low as 0.2, as shown in Fig. 2c. We found that it is an essential procedure to diminish redundancy between SSLR features for combination.

Our proposed feature refinement loss is calculated from the cross-correlation matrix $\boldsymbol{C} \in \mathbb{R}^{\mathrm{K}\times\mathrm{K}}$ between two transformed feature matrices. Thus, the optimization of the loss is equivalent to reducing correlations between input features. Using our previous notations, the cross-correlation matrix is obtained by,

$$\boldsymbol{C} = norm(\tilde{\boldsymbol{U}})^T \cdot norm(\tilde{\boldsymbol{V}}), \tag{6}$$

where $norm(\cdot)$ is the mean and *variance* normalization along time T. Note that the pre-normalization ensures the correlation is bounded between -1 and 1.

The feature refinement loss is then defined by,

$$L_{refine} \triangleq \sum_{i=1}^{K}\sum_{j=1}^{K} \begin{cases} (C_{ij})^2 & if\ |C_{ij}| > \epsilon \\ 0 & otherwise \end{cases}, \tag{7}$$

where ϵ is a threshold parameter. For example, when $\epsilon = 0.2$, any input entries that results in a correlation between -0.2 and 0.2 is masked from the loss. For batch mode training, the loss is averaged along the batch dimension.

The final loss is a combination of ASR loss and feature refinement loss with a hyperparameter λ for scaling the latter:

$$L = L_{asr} + \lambda \cdot L_{refine}. \tag{8}$$

Note that the feature refinement loss will only affect the linear projection layer during the E2E ASR model training as shown in Fig. 1, whereas the ASR loss impacts the entire architecture except the frozen pre-trained SSLR models.

4. Experiments

To evaluate the effectiveness of SSLR combinations and feature refinement loss, we first perform experiments on the Wall Street Journal (WSJ) corpus, followed by testing on Fearless Steps Challenge (FSC) phase 2 corpus. The ESPnet toolkit [21] is used for all experiments. The SSLRs used in our experiments were summarized in Sec. 2.1.

4.1. WSJ & Fearless Steps Challenge Corpora

The WSJ corpus [22] consists of 81 hours of read newspaper speech. The standard subset si284, dev93, and eval92 are used for training, development, and testing in out experiments.

The FSC phase 2 corpus [23, 24] is a subset of the 19,000-hour original Fearless Steps Corpus. FSC-2 is the original Apollo conversational speech with labeled meta-data that consists of 5 selected channels: Network Controller (NTWK), Electrical, Environmental and Consumables Manager (EECOM), Guidance Navigation and Control (GNC), Flight Director (FD), and Mission Operations Control Room (MOCR). There are distinct acoustic characteristics(e.g., noise, distortion, background inference, etc.) within these channels. In this work, we conduct experiments on ASR track 2 with segmented utterance level transcriptions. The training, development, and evaluation sets are 28 hours, 7.6 hours, and 10.6 hours respectively.

4.2. Setups

We use SSLR models with frozen parameters to extract features on the fly. Extracted features are first projected to 100 dimensions if using linear projection (Sec. 3.2.1) or weighted sum (Sec. 3.2.2). Next, features are downsampled if we are combining features with different time strides — note that the time stride for Hubert and Wav2Vec2.0 is 20ms, while all other SSLRs have a stride of 10ms. After that, features are normalized across time using statistics from each utterance and then concatenated or summed along feature dimension. The combined features are further transformed to the 80-dimensional subspace of ASR task by a linear layer. Finally, these processed features are fed into the E2E ASR model, consisting of a 12-layers Conformer [25] encoder followed by a 6-layers Transformer decoder.

When using the proposed feature refinement loss, the loss weight is set to 0.3 with the threshold parameter $\epsilon = 0.2$ for WSJ, and a loss weight 0.005 with $\epsilon = 0.6$ for FSC corpus. We use a warm-up learning scheduler. It has a peak learning rate 0.002 and warm-up steps 10,000 for WSJ while a peak learning rate 0.002 and warm-up steps 18,000 for FSC corpus. WSJ experiments use a language model (LM), while the FSC experiments do not.

Table 1: *WER of different combinations of SSLRs using concatenation on WSJ corpus.*

Model	With LM (%)		No LM (%)	
	dev93	eval92	dev93	eval92
Hubert	3.0	**1.6**	4.6	3.5
Wav2Vec2.0	3.0	2.1	4.8	3.7
TERA	6.4	4.0	10.8	8.8
CPC	6.6	3.9	11.5	8.9
APC	7.2	4.4	12.0	8.7
Hubert + Wav2Vec2.0	**2.8**	**1.6**	**4.4**	**3.4**
TERA + Wav2Vec2.0	3.2	1.8	4.8	3.6
CPC + Wav2Vec2.0	3.2	2.0	5.2	4.2
TERA + APC	6.8	3.8	11.8	8.4
CPC + APC	6.5	4.5	11.6	8.9

4.3. Results and Analysis

4.3.1. Results of Combining SSLR

In Table 1, we show word error rate (WER) performance for combinations of different SSLRs using the concatenation method on WSJ. We found that individually powerful SSLRs, such as Hubert and Wav2Vec2.0, drive the combination results. This can be seen when combining TERA with Wav2Vec2.0 instead of APC and combining CPC with Wav2Vec2.0 instead of APC. If Wav2Vec2.0 is included, the result is better. However, TERA or CPC combined with Wav2Vec2.0 only outperforms Wav2Vec2.0 in the eval92 set. When combining SSLRs of comparable strength, better results typically occur. For example, combining TERA or CPC with APC is better than using APC alone, or combining Hubert with Wav2Vec2.0 is also better than individually. In all, we obtain the best WER when combining Hubert with Wav2Vec2.0 SSLRs.

However, we also observe that adding a LM might actually reduce improvements using combination. For example, the Hubert and Wav2Vec2.0 combination improves only the dev93 set after introducing LM. Although the LM does reduce the benefit of SSLR combinations, we can still conclude that combining SSLRs can be beneficial for WER performance.

4.3.2. Combination Methods Comparison

Next, in Table 2 we compare WER performances on WSJ under different fusion methods with the Hubert and Wav2Vec2.0 combination. Both concatenation and weighted sum achieve similar performance scores, with only a slight difference on the dev93 set when a LM is not applied. However, linear projection performs worse on both dev93 and eval92 sets versus the other two methods with a LM included. Finally, adding feature refinement loss in addition to linear projection achieves 2.8% and 1.5% WER on dev93 and eval92 sets, which is a +6.67% and +6.25% relative improvement compared to Hubert only. Note that simply applying the feature refinement loss on top of the linear projection method provides a 6.67% and 11.76% relative improvement on WER. We also discover that the α in weighted sum method is 0.68 for Hubert and the β is 0.32 for Wav2Vec2.0, which matched the WER performance of these two SSLRs in Table 1, showing Hubert is a more important SSLR feature.

In Table 3, we extend our experiment to the FSC corpus. Using linear projection to combine Hubert and Wav2Vec2.0 gives +5.57% and +7.88% relative WER improvements on the eval set comparing Hubert and Wav2Vec2.0 respectively. After applying feature refinement loss, we further see an -0.32% and +0.59% relative WER improvement for dev and eval sets. These results show that our proposed feature refinement loss can be generalized to a noisy naturalistic real-world corpus.

Table 2: *WER on WSJ corpus with fusion methods applied to Hubert and Wav2Vec2.0 combination. H, W, and LP stand for Hubert, Wav2Vec2.0, and linear projection respectively.*

Model	Fusion Method	With LM (%)		No LM (%)	
		dev93	eval92	dev93	eval92
H	-	3.0	1.6	4.6	3.5
H + W	Concatenation	**2.8**	1.6	4.4	3.4
H + W	Weighted Sum	**2.8**	1.6	**4.3**	3.4
H + W	LP	3.0	1.7	-	-
H + W	LP + Refinement loss	**2.8**	**1.5**	4.4	**3.3**

Table 3: *WER of FSC corpus with fusion methods applied to Hubert and Wav2Vec2.0 combination. H, W, and LP stand for Hubert, Wav2Vec2.0, and linear projection respectively. Results here are without language model.*

Model	Fusion Method	FSC (%)	
		dev	eval
Hubert	-	33.2	35.9
Wav2Vec2.0	-	33.8	36.8
Hubert + Wav2Vec2.0	LP	**31.2**	33.9
Hubert + Wav2Vec2.0	LP + Refinement loss	31.3	**33.7**

5. Conclusion

In this paper, we have explored several combinations of SSLRs using linear projection, concatenation, and weighted sum methods in an end-to-end ASR framework. With our observation reflecting correlations between features, we further proposed a feature refinement loss for redundancy reduction when combining features. Experiments conducted on WSJ and FSC corpora show clear improvements when combining SSLRs and using the feature refinement loss. We also observed a +6.25% and +6.13% relative improvement compared to Hubert only for the WSJ eval92 and FSC eval sets. Our future work will include exploring other powerful SSLRs, combining additional features, and experimenting with alternative feature refinement approaches.

6. Acknowledgement

This project was funded, in part, by NSF-CISE Award 2016725, and partially by the University of Texas at Dallas from the Distinguished University Chair in Telecommunications Engineering held by J. Hansen. The authors would like to express our sincere thanks for valuable discussion with Huang-Cheng Chou.

7. References

[1] L. Gomez, Y. Patel, M. Rusinol, D. Karatzas, and C. Jawahar, "Self-supervised learning of visual features through embedding images into text topic spaces," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 4230–4239.

[2] A. Radford, K. Narasimhan, T. Salimans, and I. Sutskever, "Improving language understanding by generative pre-training," URL https://blog.openai.com/language-unsupervised, 2018.

[3] X. Chang, T. Maekaku, P. Guo, J. Shi, Y.-J. Lu, A. S. Subramanian, T. Wang, S.-w. Yang, Y. Tsao, H.-y. Lee *et al.*, "An explo-

ration of self-supervised pretrained representations for end-to-end speech recognition," in *2021 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*. IEEE, 2021, pp. 228–235.

[4] A. Babu, C. Wang, A. Tjandra, K. Lakhotia, Q. Xu, N. Goyal, K. Singh, P. von Platen, Y. Saraf, J. Pino *et al.*, "Xls-r: Self-supervised cross-lingual speech representation learning at scale," *arXiv preprint arXiv:2111.09296*, 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] 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] Y.-A. Chung, W.-N. Hsu, H. Tang, and J. Glass, "An unsupervised autoregressive model for speech representation learning," *Proc. Interspeech 2019*, pp. 146–150, 2019.

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

[10] R. Schluter, I. Bezrukov, H. Wagner, and H. Ney, "Gammatone features and feature combination for large vocabulary speech recognition," in *IEEE ICASSP*, 2007, p. 649–652.

[11] A. Zolnay, R. Schluter, and H. Ney, "Acoustic feature combination for robust speech recognition," in *IEEE ICASSP*, 2005.

[12] S.-J. Chen, W. Xia, and J. H. Hansen, "Scenario Aware Speech Recognition: Advancements for Apollo Fearless Steps & CHiME-4 Corpora," in *2021 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*. IEEE, 2021, pp. 289–295.

[13] P. Vieting, C. Lüscher, W. Michel, R. Schlüter, and H. Ney, "On architectures and training for raw waveform feature extraction in asr," in *2021 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*. IEEE, 2021, pp. 267–274.

[14] M. Opitz, H. Possegger, and H. Bischof, "Efficient model averaging for deep neural networks," in *Asian Conference on Computer Vision*, 2016, pp. 205–220.

[15] M. Cogswell, F. Ahmed, R. Girshick, L. Zitnick, and D. Batra, "Reducing overfitting in deep networks by decorrelating representations," *International Conference on Learning Representations (ICLR)*, 2016.

[16] E. Batlle, C. Nadeu, and J. A. Fonollosa, "Feature decorrelation methods in speech recognition. a comparative study." in *ICSLP*, 1998.

[17] K. K. Paliwal, "Decorrelated and liftered filter-bank energies for robust speech recognition," in *Sixth European conference on speech communication and technology*, 1999.

[18] Y. Bao, H. Jiang, L. Dai, and C. Liu, "Incoherent training of deep neural networks to de-correlate bottleneck features for speech recognition," in *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*. IEEE, 2013, pp. 6980–6984.

[19] X. Sun, P. Wu, and S. C. Hoi, "Face detection using deep learning: An improved faster RCNN approach," *Neurocomputing*, vol. 299, pp. 42–50, 2018.

[20] K.-H. Shih, C.-T. Chiu, J.-A. Lin, and Y.-Y. Bu, "Real-time object detection with reduced region proposal network via multi-feature concatenation," *IEEE transactions on neural networks and learning systems*, vol. 31, no. 6, pp. 2164–2173, 2019.

[21] S. Watanabe, T. Hori, S. Karita, T. Hayashi, J. Nishitoba, Y. Unno, N.-E. Y. Soplin, J. Heymann, M. Wiesner, N. Chen *et al.*, "Espnet: End-to-end speech processing toolkit," *Proc. Interspeech 2018*, pp. 2207–2211, 2018.

[22] D. B. Paul and J. Baker, "The design for the Wall Street Journal-based CSR corpus," in *Speech and Natural Language: Proceedings of a Workshop Held at Harriman, New York, February 23-26, 1992*, 1992.

[23] A. Joglekar, J. H. Hansen, M. C. Shekar, and A. Sangwan, "FEARLESS STEPS Challenge (FS-2): Supervised Learning with Massive Naturalistic Apollo Data," *Proc. Interspeech 2020*, pp. 2617–2621, 2020.

[24] J. H. Hansen, A. Sangwan, A. Joglekar, A. E. Bulut, L. Kaushik, and C. Yu, "Fearless Steps: Apollo-11 Corpus Advancements for Speech Technologies from Earth to the Moon." in *INTERSPEECH*, 2018, pp. 2758–2762.

[25] A. Gulati, J. Qin, C.-C. Chiu, N. Parmar, Y. Zhang, J. Yu, W. Han, S. Wang, Z. Zhang, Y. Wu *et al.*, "Conformer: Convolution-augmented Transformer for Speech Recognition," *Proc. Interspeech 2020*, pp. 5036–5040, 2020.