

k2SSL: A Faster and Better Framework for Self-Supervised Speech Representation Learning

Yifan Yang^{1†}, Jianheng Zhuo^{1†}, Zengrui Jin³, Ziyang Ma¹, Xiaoyu Yang², Zengwei Yao²
Liyong Guo², Wei Kang², Fangjun Kuang², Long Lin², Daniel Povey², Xie Chen^{1*}

¹*MoE Key Lab of Artificial Intelligence, X-LANCE Lab, Shanghai Jiao Tong University*

²*Xiaomi Corporation, Beijing, China*

³*The Chinese University of Hong Kong, Hong Kong SAR, China*

{yifanyeung, zzasdf, chenxie95}@sjtu.edu.cn

Abstract—Self-supervised learning (SSL) has achieved great success in speech-related tasks. While Transformer and Conformer architectures have dominated SSL backbones, encoders like Zipformer, which excel in automatic speech recognition (ASR), remain unexplored in SSL. Concurrently, inefficiencies in data processing within existing SSL training frameworks, such as fairseq, pose challenges in managing the growing volumes of training data. To address these issues, we propose k2SSL, an open-source framework that offers faster, more memory-efficient, and better-performing self-supervised speech representation learning, focusing on downstream ASR tasks. The optimized HuBERT and proposed Zipformer-based SSL systems exhibit substantial reductions in both training time and memory usage during SSL training. Experiments on LibriSpeech demonstrate that Zipformer Base significantly outperforms HuBERT and WavLM, achieving up to a 34.8% relative WER reduction compared to HuBERT Base after fine-tuning, along with a 3.5x pre-training speedup in GPU hours. When scaled to 60k hours of LibriLight data, Zipformer Large exhibits remarkable efficiency, matching HuBERT Large’s performance while requiring only 5/8 pre-training steps.

Index Terms—self-supervised learning, speech recognition, memory-efficient, Zipformer

I. INTRODUCTION

Speech-based self-supervised learning (SSL) [1]–[7] has emerged as a powerful paradigm for its generalization ability, leveraging vast amounts of unlabeled data to derive universal representations. These models can be easily fine-tuned by incorporating prediction layers for specific tasks, enhancing performance in applications like automatic speech recognition (ASR) [8]. HuBERT [3], one widespread self-supervised speech representation model, utilizes the classical and versatile Transformer [9] architecture and masked language modeling (MLM), achieving generalizability across various downstream tasks with remarkable performance [10].

However, training speech-based SSL models demands enormous computational resources due to the need for massive quantities of unlabeled data, often more than ten times the amount of labeled data, for effective pre-training before fine-tuning. For instance, training HuBERT Base requires 32 GPUs, taking 24 hours for the first iteration and 38 hours for the second iteration. When scaling up to larger variants, the requirement jumps to 128 GPUs for HuBERT Large and 256

GPUs for HuBERT X-Large. Similarly, WavLM [4] training demands a lot of computational resources. WavLM Base needs 32 GPUs, while the WavLM Large requires 64 GPUs. The extensive memory usage and prolonged training time of these models make them inaccessible to most researchers, contradicting the fast-paced demands of the AI industry.

On the other hand, datasets used in both industry and academia have grown prohibitively large [11]–[18], ranging from tens of thousands to hundreds of thousands of hours. These massive datasets have surpassed the capabilities of existing open-source speech-based SSL frameworks [19]–[21], which struggle with excessive memory demands and inefficient data management when handling such large-scale data. Consequently, there is considerable room for improving the accessibility, efficiency, and effectiveness of speech-based SSL systems.

With this perspective in mind, we propose k2SSL¹, an open-source framework designed to provide a faster, more memory-efficient, and higher-performing solution for self-supervised speech representation learning. k2SSL specifically addresses the inefficiencies encountered by existing frameworks when managing large-scale datasets, offering seamless scaling for both storage and training.

We optimize the HuBERT architecture by removing memory-intensive components that contribute little to performance, and we integrate the Zipformer [22] encoder as the backbone, combined with the ScaledAdam [22] optimizer. Zipformer Base pre-training requires only 8 V100 32G GPUs, while Zipformer Large requires 32 V100 32G GPUs, both without gradient accumulation, significantly reducing computational demands. This makes k2SSL more accessible and feasible for a broader range of research teams and organizations. Extensive experiments on LibriSpeech [23] and LibriLight [24] demonstrate that Zipformer-based SSL systems exhibit strong performance and efficiency. Specifically, Zipformer Base achieves a relative WER reduction of up to 34.8%/32.4% on dev-other/test-other compared to HuBERT Base after supervised fine-tuning, while delivering a 3.5x pre-training speedup in total GPU hours, highlighting both efficiency and effectiveness. Zipformer Large demonstrates

[†]Equal contribution. *Corresponding author.

¹Code and models are available at <https://github.com/k2-fsa/icefall>.

exceptional efficiency, achieving performance comparable to HuBERT Large while requiring only 5/8 of the pre-training steps.

II. HUBERT AND ITS VARIANTS

A. HuBERT

HuBERT, short for Hidden-Unit BERT, is built on the wav2vec 2.0 [2] architecture and incorporates a convolutional feature extractor, a Transformer encoder, and a linear projection layer. It processes raw speech and predicts k -means clustering labels for masked audio segments. HuBERT processes audio at a standard 16kHz rate, downsampling the sequences to 50Hz through the convolutional feature extractor. A predetermined percentage of timestamps are selected as starting points for mask spans, covering multiple consecutive timestamps to form the masked sections of the sequence. For these masked parts, the speech features are replaced with a learnable mask embedding. The model then calculates the probability distribution across cluster labels using the following equation:

$$p_f(c|\tilde{X}, t) = \frac{\exp(\text{sim}(A o_t, e_c) / \tau)}{\sum_{i=1}^C \exp(\text{sim}(A o_t, e_i) / \tau)} \quad (1)$$

Here, \tilde{X} represents the masked input, $[o_1, \dots, o_T]$ the output from the Transformer encoder, A the projection matrix, e_c the cluster label embedding, C the number of clusters, and τ the logit scaler, set to a specific value. HuBERT Base is pre-trained in two iterations. The first iteration uses 39-dimensional MFCC features from the LibriSpeech-960h dataset, clustered with k-means using a codebook size of 100. In the second iteration, features from the 6th layer of the first model are clustered with a codebook size of 500. HuBERT Large is pre-trained on clusters derived from the 9th layer of the second iteration, again using a codebook size of 500.

Several follow-up works [4], [25]–[32] have been proposed to boost either efficiency or effectiveness based on HuBERT.

B. Efficient Variants

MelHuBERT [26] simplifies HuBERT training by using Fbank as input and replacing the pre-train loss with the cross-entropy. Fast-HuBERT [27] also accelerates HuBERT training by utilizing Fbank features with larger frameshifts as input. However, using larger frameshift Fbank features requires extracting them in advance, which introduces extra time and storage costs and leads to a notable decline in performance.

C. Effective Variants

HuBERT-AP [28] and CTCBERT [30] change how the Transformer output is aligned with the SSL units, making the pre-training process more like the common practice of speech recognition. Nonetheless, these implementation modifications have limited performance improvement over the original HuBERT. PBERT [31], MonoBERT [32], and PolyBERT [32] introduce phoneme information to the SSL units to improve performance. However, a pre-trained phoneme recognizer with

pair data is needed in PBERT, and additional wav2vec-U 2.0 [33] is required in MonoBERT and PolyBERT. WavLM [4] effectively enhances the multi-task performance by increasing the data scale and incorporating additional denoising loss during the pre-training stage, but it requires a substantial computational cost. Boosting efficiency and effectiveness simultaneously in speech-based SSL models remains to be explored.

III. k2SSL

k2SSL aims to enhance both the efficiency and effectiveness of HuBERT-based SSL systems. We begin by analyzing design inefficiencies in these systems, which lead to unnecessary computational overhead, and present our corrective modifications. We then describe our transition from the Transformer encoder to Zipformer and the integration of the ScaledAdam optimizer. Finally, we outline several engineering advantages of the proposed k2SSL framework. The overall SSL system, with Zipformer as the backbone, is depicted in Fig. 1.

A. Analysis and Optimization of HuBERT Architecture

HuBERT incorporates the convolutional feature extractor from the wav2vec 2.0 architecture to process audio sampled at 16kHz, which includes 7-layer convolutional layers. In HuBERT Base, an Fp32 GroupNorm is applied after the first convolution, while in HuBERT Large, an Fp32 LayerNorm is applied after each convolution. Testing on an idle V100 GPU with a batch of six 60-second audio samples showed peak memory usage for the convolutional feature extractor: 10.96 GB for HuBERT Base and 13.14 GB for HuBERT Large, reflecting a 19.9% increase in memory overhead. Our experiments indicate that although applying Fp32 LayerNorm after each layer enhances stability during half-precision training, it contributes little to performance and incurs huge memory overhead. Instead, we retain a single Fp32 GroupNorm in the first layer and resort to the half-precision stable Zipformer + ScaledAdam framework.

Following [26], cross-entropy loss is used for pre-training, largely reducing peak memory usage by eliminating updates to codeword embeddings and cosine similarity calculations. This simplified loss function predicts the k -means clustering labels of the masked input \tilde{X} as follows:

$$p_f(c|\tilde{X}, t) = \frac{\exp((A o_t)_c)}{\sum_{i=1}^C \exp((A o_t)_i)}, \quad (2)$$

where o_t denotes the Transformer encoder output at time step t , and A represents the projection matrix.

B. Adapting Zipformer and ScaledAdam for SSL

The Transformer encoder exhibits quadratic memory and computational complexity as the input sequence length increases. However, self-supervised learning benefits more from longer sequences, creating a conflict between efficiency and performance. Furthermore, as noted in [3], the total batch size is a critical factor influencing model performance.

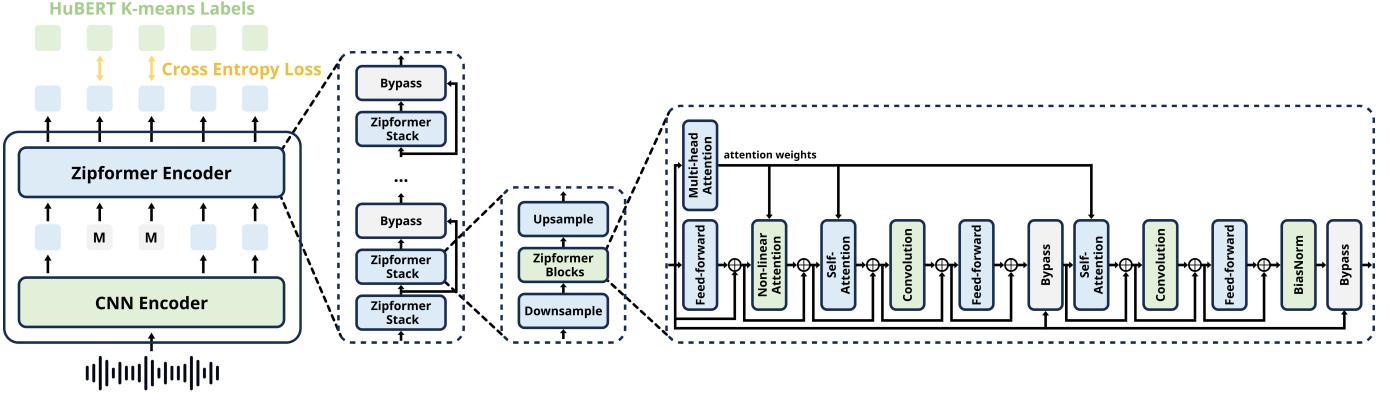


Fig. 1: The overall architecture of the SSL system with Zipformer as the backbone.

To tackle these, we transition to Zipformer, which handles long sequences more memory efficiently and supports larger batch sizes, enabling more efficient and effective SSL pre-training. As shown in Fig. 1, Zipformer [22] encoder adopts a U-Net-like structure to learn temporal representations efficiently at varying resolutions across different stacks. Starting with an acoustic feature sequence at a 50Hz frame rate, Zipformer processes it through six cascaded stacks, with frame rates of 50Hz, 25Hz, 12.5Hz, 6.25Hz, 12.5Hz, and 25Hz. Except for the first stack, each downsamples the sequence, processes it, and then upsamples it back to 50Hz. The embedding dimensions vary across the stacks, with the central stacks having larger dimensions. The output from each stack is truncated or padded with zeros to match the dimensions of the subsequent stack. The final encoder output is set to the maximum dimension across all stacks, concatenating segments from each stack’s output and taking the most recent output for each dimension. Zipformer also incorporates several upgrades: a redesigned block structure that reuses attention weights for efficiency, BiasNorm [22] for better retention of sequence length, and activation functions SwooshR [22] and SwooshL [22], surpassing the performance of Swish. Additionally, ScaledAdam [22], a parameter-scale-invariant version of Adam, decouples gradient scale and direction, further improving convergence.

C. Fine-tuning with Pruned RNN-T Loss

Letter-level CTC [34] loss is commonly employed for downstream ASR in speech-based SSL, but it is inefficient, slow to converge, and hard to align. In addition to using letter-level CTC loss for fair comparisons, we leverage pruned RNN-T loss [35], a memory-efficient variant of transducer loss [36], to enhance ASR fine-tuning effectiveness with minimal overhead.

D. Technical Advantages of k2SSL

k2SSL integrates Lhotse [37] to efficiently manage large-scale datasets by optimizing I/O bandwidth and storage capacity, enabling seamless scaling for both storage and training. It excels in processing speech segments stored in shuffled order

TABLE I: Hyperparameters of various models in k2SSL for pre-training and fine-tuning. Base models are pre-trained on 8 NVIDIA V100 32GB GPUs, and Large models are on 32 NVIDIA V100 32GB GPUs. All models are fine-tuning on 8 NVIDIA V100 32GB GPUs. “LR” represents the learning rate, “BS” represents the duration of speech samples in a single batch, and “GA” represents gradient accumulation.

Backbone	Stage		Loss	# Params	LR	BS/GPU	GA
	Pre-train	Fine-tune					
Transformer	✓		HuBERT	94.7M	0.045	87.5s	4
		✓	Pruned RNN-T	96.5M	0.001	200.0s	1
Transformer	✓		CE	94.8M	0.045	87.5s	4
		✓	Pruned RNN-T	96.6M	0.00075	200.0s	1
Zipformer	✓		CE	94.6M	0.045	600.0s	1
	✓	✓	Pruned RNN-T	96.4M	0.002	600.0s	1
Zipformer	✓		CTC	94.6M	0.001	600.0s	1
		✓	CE	305.9M	0.045	350.0s	1
		✓	Pruned RNN-T	308.0M	0.002	400.0s	1

by duration, utilizing the `DynamicBucketingSampler` from Lhotse to eliminate the need to load entire manifests into memory. By streamingly reading a subset of manifests to estimate bucket boundaries and maintaining a fixed-size buffer, this approach ensures consistent data loading times, regardless of dataset size. As a result, pre-training on vast datasets like the 60,000-hour Libri-Light [24] is initiated in merely dozens of seconds, in contrast to the lengthy loading times and huge memory demands of traditional frameworks.

Moreover, pre-trained HuBERT checkpoints from fairseq² are compatible with k2SSL, supporting further fine-tuning for downstream tasks.

IV. EXPERIMENTS

A. Experimental Setups

Four distinct experimental setups are detailed in Table I, with all Base models containing ($\sim 95\text{M}$) parameters and Large

²<https://github.com/facebookresearch/fairseq/tree/main/examples/hubert>

TABLE II: Comparison of Word Error Rates (WERs) **without an external language model** during decoding on LibriSpeech dev/test sets and pre-training speedup ratios measured in total GPU hours. All models are pre-trained on unlabeled **LibriSpeech-960h** and fine-tuned on labeled **LibriSpeech-100h** using pruned RNN-T loss, 500-class BPE word pieces as modeling units, and ScaledAdam as optimizer.

Model	Backbone	Pre-train			Word Error Rate (%)				GPU Hours	Pre-train Speedup
		Toolkit	Optimizer	Loss Function	dev-clean	dev-other	test-clean	test-other		
HuBERT Base	Transformer	fairseq	Adam	HuBERT	4.88	12.06	4.98	11.65	1878	1x
HuBERT Base	Transformer	k2SSL	ScaledAdam	HuBERT	4.67 _{-4.3%}	12.05 _{-0.0%}	4.81 _{-3.4%}	11.53 _{-1.0%}	1270	1.48x
HuBERT Base	Transformer	k2SSL	ScaledAdam	CE	4.54 _{-7.0%}	11.61 _{-3.7%}	4.77 _{-4.2%}	11.09 _{-4.8%}	886	2.12x
Zipformer Base	Zipformer	k2SSL	ScaledAdam	CE	3.67 _{-24.8%}	7.86 _{-34.8%}	3.80 _{-23.7%}	7.87 _{-32.4%}	531	3.53x

model containing ($\sim 306M$) parameters. Setups 1 and 2 (Rows 1 & 2) utilize the Transformer encoder as the backbone. Setup 1 uses the original HuBERT loss, and Setup 2 adopts cross-entropy (CE) loss during pre-training. Setups 3 and 4 (Rows 3 & 4) employ Zipformer Base and Zipformer Large with CE loss for pre-training. Setups 1, 3, and 4 set batch sizes to fully exploit single GPU memory. With the memory-efficient designs, Zipformer Base and Zipformer Large require only 8 and 32 GPUs, respectively, without gradient accumulation. This allows for a larger effective batch size while requiring only 1/4 of the GPUs compared to the original HuBERT [3], which necessitates 32 and 128 GPUs.

Dataset 960 hours of LibriSpeech [23] audio or 60,000 hours of Libri-light [24] audio are used for pre-training. 100 hours or 960 hours of LibriSpeech are considered for fine-tuning.

Pre-training We directly utilize the target labels identical to those used in HuBERT [3] and WavLM [4] for fair comparisons. Since these targets are well-established, there is no need for metrics like phone purity, cluster purity, or PNMI, aligning with our intention to attribute performance differences to model and training variations rather than target quality. All Base models are pre-trained for 300 epochs (400k steps for HuBERT Base, 225k steps for Zipformer Base) on unlabeled 960 hours of LibriSpeech audio, using 500-class k-means labels³ derived from clustering the 6th layer of the first iteration HuBERT Base model. Zipformer Large is pre-trained for 250k steps on 60,000 hours of Libri-Light, using 500-class k-means labels⁴ extracted from the 9th layer of the second iteration HuBERT Base model⁵. The same masking strategies as in [3] are used. ScaledAdam with $\beta = (0.9, 0.98)$ is applied alongside Eden [22] scheduler.

Fine-tuning with Pruned RNN-T Loss 500-class Byte Pair Encoding (BPE) [38] word pieces are selected as the classification units. For decoding, we adopt constrained beam search [39] with a beam size of 8, which permits only one symbol emission per frame.

Fine-tuning with CTC Loss The CTC vocabulary comprises 26 English letters, a space, an apostrophe, and a CTC blank symbol. The decoding process with an external language

TABLE III: Comparison of WERs w/wo a language model between Zipformer Base and baseline models on LibriSpeech dev/test sets. All models are pre-trained on unlabeled **LibriSpeech-960h** and fine-tuned on labeled **LibriSpeech-100h**, using CTC loss and letters as modeling units. The results of baseline models are collected from their respective papers or fine-tuned based on official checkpoints.

Model	LM	Word Error Rate (%)			
		dev-clean	dev-other	test-clean	test-other
wav2vec 2.0 Base [2]	None	6.1	13.5	6.1	13.3
HuBERT Base	None	5.3	13.0	5.4	12.6
WavLM Base [4]	None	-	-	5.7	12.0
Zipformer Base	None	4.7	9.6	4.4	9.8
wav2vec 2.0 Base [2]	4-gram	2.7	7.9	3.4	8.0
HuBERT Base [3]	4-gram	2.7	7.8	3.4	8.1
WavLM Base [4]	4-gram	-	-	3.4	7.7
Zipformer Base	4-gram	2.6	6.4	3.0	6.8

model follows [3], utilizing the wav2letter++ [40] beam search decoder, formulated as follows:

$$\log p_{CTC}(\mathbf{y} \mid \mathbf{x}) + w_1 \log p_{LM}(\mathbf{y}) + w_2 |\mathbf{y}|, \quad (3)$$

where \mathbf{x} is the input audio, \mathbf{y} is the predicted text, $|\mathbf{y}|$ is length of the text, and w_1 and w_2 are the language model weight and word score coefficients.

B. Experimental Results

Table II presents the ASR performance of the fairseq pre-trained HuBERT Base (Row 1), k2SSL pre-trained HuBERT Base with two distinct pre-training losses (Rows 2 & 3), and k2SSL pre-trained Zipformer Base (Row 4). All models are fine-tuned within the k2SSL framework on the LibriSpeech-100h, using the pruned RNN-T loss. The following conclusions can be drawn from the analysis:

- Compared to the fairseq pre-trained HuBERT Base, the k2SSL pre-trained HuBERT Base with the assistance of ScaledAdam optimizer achieves a speedup ratio of approximately 1.5 (Row 2 vs. 1);

³librispeech-hubert-base-ls960-iter1-l6-km500

⁴librilight-hubert-base-ls960-iter2-l9-km500

⁵https://dl.fbaipublicfiles.com/hubert/hubert_base_ls960.pt

TABLE IV: Comparison of WERs **without an external language model** on LibriSpeech test sets. All models are **trained with transducer or pruned RNN-T loss on labeled LibriSpeech-960h**. All pre-trained models are compared based on the number of pre-training steps required to achieve similar performance, rather than maximizing performance.

Model	Unlabeled Data	Pre-train Steps	Word Error Rate (%)	
			test-clean	test-other
Supervised				
Transformer Transducer [41]	-	-	2.4	5.6
Conformer-L Transducer [42]	-	-	2.1	4.3
Conformer-L Pruned Transducer [22]	-	-	2.5	5.6
Zipformer-L Pruned Transducer [22]	-	-	2.1	4.6
Pre-trained				
Conformer-L [43]	LL-60k	400k	2.0	4.5
HuBERT Large	LL-60k	400k	1.8	3.9
Zipformer Large	LL-60k	250k	1.8	4.0

- The simplified pre-training loss further provides a around 2.1x pre-training speedup and a relative WER reduction of up to 3.8% (Row 3 vs. 2, test-other);
- When using Zipformer as the backbone, the largest WER reduction is obtained across all dev and test sets of LibriSpeech (Row 4 vs. 3). Compared to the original HuBERT Base, Zipformer Base achieves significant relative WER reductions by up to 24.8% on dev-clean, 34.8% on the dev-other, 23.7% on test-clean, and 32.4% on test-other while yielding a 3.53 pre-training speedup (Row 4 vs. 1). This indicates the superiority of the k2SSL framework with the Zipformer encoder as the backbone.

Table III presents the ASR performance of Zipformer Base against three leading SSL models. The results are either collected from the original paper or fine-tuned on LibriSpeech-100h using CTC loss with letter-level modeling units based on publicly available checkpoints. The key observations from the results can be made:

- Without language model fusion, Zipformer Base significantly outperforms wav2vec 2.0 Base, HuBERT Base, and WavLM Base by relative WER reductions up to 28.9% (Row 4 vs. 1, dev-other), 26.2% (Row 4 vs. 2, dev-other), and 22.8% (Row 4 vs. 3, test-clean), respectively;
- The performance advancement of Zipformer Base is maintained when incorporating an external 4-gram language model for decoding. Zipformer Base brings a relative WER reduction of up to 19.0% (Row 8 vs. 5, dev-other), 18.0% (Row 8 vs. 6, dev-other), and 11.8% (Row 8 vs. 7, test-clean), respectively.

Table IV reports the ASR performance of Large variants on the full LibriSpeech. Results are either sourced from the original paper or fine-tuned on LibriSpeech-960h within the k2SSL framework, using pruned RNN-T loss with publicly available checkpoints. The main takeaways from the results are:

- Both HuBERT Large and Zipformer Large, fine-tuned within the k2SSL framework, outperform all supervised

models;

- Zipformer Large (308.0M) achieves comparable performance against HuBERT Large (318.7M) with much fewer pre-training steps (250k vs. 400k). We are optimistic that more pre-training steps will yield better results, as observed in the Base variant.

These observations show the remarkable capabilities and advancements of the k2SSL approach in improving both efficiency and effectiveness.

V. CONCLUSIONS

This paper introduces k2SSL, a scalable framework designed for more efficient and effective self-supervised speech representation learning, tailored for downstream ASR tasks. By optimizing the HuBERT architecture and adopting Zipformer as the backbone, k2SSL achieves remarkable reductions in both memory usage and training time, making it accessible to a broader range of research teams and organizations. Extensive experiments on LibriSpeech and Libri-Light validate the superiority of k2SSL, with Zipformer-based SSL systems significantly outperforming HuBERT and WavLM, achieving a 3.5x pre-training speedup and notable WER reductions. The related resources, including the k2SSL framework, training configurations, and pre-trained models, are publicly available to facilitate further research.

ACKNOWLEDGMENT

This work was supported by the National Natural Science Foundation of China (No. U23B2018 and No. 62206171), Shanghai Municipal Science and Technology Major Project under Grant 2021SHZDZX0102 and the International Cooperation Project of PCL.

REFERENCES

- [1] S. Schneider, A. Baevski, R. Collobert, and M. Auli, “wav2vec: Unsupervised pre-training for speech recognition,” in *Proc. INTERSPEECH*, Graz, 2019.

- [2] A. Baevski, Y. Zhou, A. Mohamed, and M. Auli, “wav2vec 2.0: A framework for self-supervised learning of speech representations,” in *Proc. NeurIPS*, 2020.
- [3] W. N. Hsu, B. Bolte, Y. H. H. Tsai, K. Lakhota, 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, 2021.
- [4] S. Chen, C. Wang, Z. Chen *et al.*, “WavLM: Large-scale self-supervised pre-training for full stack speech processing,” *IEEE Journal of Selected Topics in Signal Processing*, vol. 16, 2022.
- [5] A. Baevski, A. Babu, W. N. Hsu, and M. Auli, “data2vec: A general framework for self-supervised learning in speech, vision and language,” in *Proc. ICML*, Baltimore, 2022.
- [6] ———, “Efficient self-supervised learning with contextualized target representations for vision, speech and language,” in *Proc. ICML*, Honolulu, 2023.
- [7] A. Mohamed, H. yi Lee, L. Borgholt *et al.*, “Self-supervised speech representation learning: A review,” *IEEE Journal of Selected Topics in Signal Processing*, vol. 16, 2022.
- [8] J. Li *et al.*, “Recent advances in end-to-end automatic speech recognition,” *APSIPA Transactions on Signal and Information Processing*, vol. 11, 2022.
- [9] A. Vaswani, N. Shazeer, N. Parmar *et al.*, “Attention is all you need,” in *Proc. NeurIPS*, Long Beach, 2017.
- [10] S. W. Yang, H. Chi, Y. S. Chuang *et al.*, “SUPERB: Speech processing universal performance benchmark,” in *Proc. INTERSPEECH*, Brno, 2021.
- [11] M. Łajszczak, G. Cámbara, Y. Li *et al.*, “BASE TTS: Lessons from building a billion-parameter text-to-speech model on 100k hours of data,” *arXiv preprint arXiv:2402.08093*, 2024.
- [12] M. Le, A. Vyas, B. Shi *et al.*, “Voicebox: Text-guided multilingual universal speech generation at scale,” in *Proc. NeurIPS*, Vancouver, 2024.
- [13] V. Pratap, A. Tjandra, B. Shi *et al.*, “Scaling speech technology to 1,000+ languages,” *arXiv preprint arXiv:2305.13516*, 2023.
- [14] A. Radford, J. W. Kim, T. Xu *et al.*, “Robust speech recognition via large-scale weak supervision,” in *Proc. ICML*, Honolulu, 2023.
- [15] Y. Zhang, W. Han, J. Qin *et al.*, “Google USM: Scaling automatic speech recognition beyond 100 languages,” *arXiv preprint arXiv:2303.01037*, 2023.
- [16] W. Kang, X. Yang, Z. Yao, F. Kuang, Y. Yang, L. Guo, L. Lin, and D. Povey, “Libriheavy: a 50,000 hours ASR corpus with punctuation casing and context,” in *Proc. ICASSP*, Seoul, 2024.
- [17] Y. Yang, Z. Song, J. Zhuo *et al.*, “Gigaspeech 2: An evolving, large-scale and multi-domain ASR corpus for low-resource languages with automated crawling, transcription and refinement,” *arXiv preprint arXiv:2406.11546*, 2024.
- [18] Z. Jin, Y. Yang, M. Shi, W. Kang, X. Yang, Z. Yao, F. Kuang, L. Guo, L. Meng, L. Lin, Y. Xu, S.-X. Zhang, and D. Povey, “LibriheavyMix: a 20,000-hour dataset for single-channel reverberant multi-talker speech separation, ASR and speaker diarization,” in *Proc. INTERSPEECH*, Kos Island, 2024.
- [19] M. Ott, S. Edunov, A. Baevski *et al.*, “fairseq: A fast, extensible toolkit for sequence modeling,” in *Proc. NAACL-HLT*, Minneapolis, 2019.
- [20] S. Watanabe, T. Hori, S. Karita *et al.*, “ESPnet: End-to-end speech processing toolkit,” in *Proc. INTERSPEECH*, Hyderabad, 2018.
- [21] A. T. Liu and Y. S. wen, “The S3PRL toolkit: Self-supervised speech pre-training and representation learning,” <https://github.com/s3prl/s3prl>, 2020.
- [22] Z. Yao, L. Guo, X. Yang, W. Kang, F. Kuang, Y. Yang, Z. Jin, L. Lin, and D. Povey, “Zipformer: A faster and better encoder for automatic speech recognition,” in *Proc. ICLR*, Vienna, 2024.
- [23] V. Panayotov, G. Chen, D. Povey, and S. Khudanpur, “Librispeech: an ASR corpus based on public domain audio books,” in *Proc. ICASSP*, South Brisbane, 2015.
- [24] J. Kahn, M. Rivière, W. Zheng *et al.*, “Libri-Light: A benchmark for ASR with limited or no supervision,” in *Proc. ICASSP*, Barcelona, 2020.
- [25] W. Chen, X. Chang, Y. Peng, Z. Ni, S. Maiti, and S. Watanabe, “Reducing barriers to self-supervised learning: HubERT pre-training with academic compute,” in *Proc. INTERSPEECH*, Dublin, 2023.
- [26] T. Q. Lin, H. Y. Lee, and H. Tang, “MeHubERT: A simplified hubert on mel spectrograms,” in *Proc. ASRU*, Taipei, 2023.
- [27] G. Yang, Z. Ma, Z. Zheng *et al.*, “Fast-HuBERT: an efficient training framework for self-supervised speech representation learning,” in *Proc. ASRU*, Taipei, 2023.
- [28] S. Ren, S. Liu, Y. Wu, L. Zhou, and F. Wei, “Speech pre-training with acoustic piece,” in *Proc. INTERSPEECH*, Incheon, 2022.
- [29] Y. Yang, F. Shen, C. Du, Z. Ma, K. Yu, D. Povey, and X. Chen, “Towards universal speech discrete tokens: A case study for ASR and TTS,” in *Proc. ICASSP*, Seoul, 2024.
- [30] R. Fan, Y. Wang, Y. Gaur, and J. Li, “Ctcbert: Advancing hidden-unit bert with ctc objectives,” in *Proc. ICASSP*, Rhodes Island, 2023.
- [31] C. Wang, Y. Wang, Y. Wu, S. Chen, J. Li, S. Liu, and F. Wei, “Supervision-guided codebooks for masked prediction in speech pre-training,” in *Proc. INTERSPEECH*, Incheon, 2022.
- [32] Z. Ma, Z. Zheng, G. Yang, Y. Wang, C. Zhang, and X. Chen, “Pushing the limits of unsupervised unit discovery for SSL speech representation,” in *Proc. INTERSPEECH*, Dublin, 2023.
- [33] A. H. Liu, W. N. Hsu, M. Auli, and A. Baevski, “Towards end-to-end unsupervised speech recognition,” in *Proc. SLT*, Doha, 2022.
- [34] A. Graves, S. Fernández, F. J. Gomez, and J. Schmidhuber, “Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks,” in *Proc. ICML*, Pittsburgh, 2006.
- [35] F. Kuang, L. Guo, W. Kang, L. Lin, M. Luo, Z. Yao, and D. Povey, “Pruned RNN-T for fast, memory-efficient ASR training,” in *Proc. INTERSPEECH*, Incheon, 2022.
- [36] A. Graves, A. rahman Mohamed, and G. E. Hinton, “Speech recognition with deep recurrent neural networks,” in *Proc. ICASSP*, Vancouver, 2013.
- [37] P. Zelasko, D. Povey, J. Trmal, and S. Khudanpur, “Lhotse: a speech data representation library for the modern deep learning ecosystem,” in *NeurIPS Data-Centric AI Workshop*, 2021.
- [38] R. Sennrich, B. Haddow, and A. Birch, “Neural machine translation of rare words with subword units,” in *Proc. ACL*, Berlin, 2016.
- [39] W. Kang, L. Guo, F. Kuang, L. Lin, M. Luo, Z. Yao, X. Yang, P. Zelasko, and D. Povey, “Fast and parallel decoding for transducer,” in *Proc. ICASSP*, Rhodes Island, 2023.
- [40] V. Pratap, A. Y. Hannun, Q. Xu *et al.*, “Wav2letter++: A fast open-source speech recognition system,” in *Proc. ICASSP*, Brighton, 2019.
- [41] Q. Zhang, H. Lu, H. Sak, A. Tripathi, E. McDermott, S. Koo, and S. Kumar, “Transformer transducer: A streamable speech recognition model with transformer encoders and RNN-T loss,” in *Proc. ICASSP*, Barcelona, 2020.
- [42] A. Gulati, J. Qin, C. C. Chiu *et al.*, “Conformer: Convolution-augmented transformer for speech recognition,” in *Proc. INTERSPEECH*, Shanghai, 2020.
- [43] Y. Zhang, J. Qin, D. S. Park *et al.*, “Pushing the limits of semi-supervised learning for automatic speech recognition,” in *Proc. NeurIPS Self-Supervised Learning for Speech and Audio Processing Workshop*, 2020.