Interspeech 2025
17-21 August 2025, Rotterdam, The Netherlands

Improving Multilingual Speech Models on ML-SUPERB 2.0: Fine-tuning with Data Augmentation and LID-Aware CTC

Qingzheng Wang, Jiancheng Sun*, Yifan Peng, Shinji Watanabe*

Carnegie Mellon University, USA

{qingzhew, jianches, yifanpen}@andrew.cmu.edu, shinjiw@ieee.org

Abstract

Multilingual speech processing with self-supervised or supervised pre-trained Speech Foundation Models (SFM) has achieved strong performance on tasks like Language Identification (LID) and Automatic Speech Recognition (ASR). However, these models struggle with limited resources during fine-tuning. This paper enhances multilingual LID and ASR on ML-SUPERB 2.0 by exploring multiple strategies for adapting SFMs, including frozen upstream training, partial fine-tuning, and low-rank adaptation. Furthermore, we employ data augmentation to mitigate performance gaps in few-shot settings and introduce LID Connectionist Temporal Classification (CTC) loss for regularization. Our approach achieves a 14% relative improvement in LID accuracy and a 30% relative reduction in ASR CER over the baseline on ML-SUPERB 2.0, securing second place in the Interspeech 2025 ML-SUPERB 2.0 Challenge.
Index Terms: multilingual speech recognition, language identification, efficient fine-tuning, data augmentation.

1. Introduction

Multilingual speech processing, including tasks such as Language Identification (LID) and Automatic Speech Processing (ASR), is essential in a world with more than 7000 languages [1, p. 1]. Recent advancements in multilingual speech foundation models (SFM) based on self-supervised learning (SSL) have significantly expanded the scope of robust multilingual speech processing capabilities to at most over 4000 languages with large-scale unlabeled data [2–7]. In parallel, the supervised pre-trained SFM has also advanced, enabling scalable solutions across over 100 languages and multiple tasks [8–11].

To evaluate SFM in multilingual LID and ASR, ML-SUPERB 2.0 [12] is introduced as an enhanced benchmark, covering over 140 languages and 56 dialects. Unlike ML-SUPERB [13], which restricted evaluation to frozen upstream parameters and fixed architectures, ML-SUPERB 2.0 adopts a more flexible framework without strategy restrictions. This flexibility enables broader evaluation across diverse models, strategies, and learning scenarios.

While various strategies and models have been evaluated on ML-SUPERB 2.0, prior work reveals significant deviations in LID and ASR performance across languages [12]. Notably, both performances degrade substantially for languages under few-shot training conditions [12]. Another key challenge is that fine-tuning on low-resource languages with limited labeled data leads to overfitting and poor generalization to unseen dialects.

In this paper, we propose to improve the performance of multilingual LID and ASR on ML-SUPERB 2.0. Specifically, our work focuses on three key aspects: First, we investigate training strategies for adapting SFMs on ML-SUPERB 2.0, including frozen upstream training, partial fine-tuning, and Low-Rank Adaptation (LoRA). Second, to improve performance on low-resource languages, we incorporate additional labeled data into the training data. Third, to regularize fine-tuning, we employ an auxiliary Connectionist Temporal Classification (CTC) [14–18] loss for LID prediction during SFM fine-tuning. This enhances generalization to unseen dialects. Our approach achieves an LID accuracy of 86.9% and an ASR Character Error Rate (CER) of 15.6% on ML-SUPERB 2.0.

*These authors contributed equally.

2. Related studies

Multilingual speech foundation models. Both supervised and self-supervised training strategies are used to build multilingual SFMs. Self-supervised pre-trained SFMs like Massively Multilingual Speech (MMS) [5] and XEUS [6], trained on 1400+ and 4000+ languages respectively, achieve remarkable multilingual performance. Meanwhile, supervised pre-trained SFMs such as encoder-decoder-based Whisper [9] (99 languages) and encoder-only OWSM-CTC [11] (151 languages) rely on labeled speech during training. In this paper, we conduct evaluations of three representative encoder-only multilingual SFMs (MMS, XEUS, and OWSM-CTC) on ML-SUPERB 2.0.

Data augmentation. Prior work has explored enhancing ASR model performance on unknown languages by incorporating additional language data for training [19]. Another work [20] enhances low-resource ASR by leveraging self-training or a text-to-speech system to generate more training data. Speed perturbation [21] and SpecAugment [22] augment audio or feature inputs to improve the training data quality. In contrast, our work focuses on improving underperforming languages utilizing additional labeled data from public datasets.

LID-aware training. The intermediate CTC loss has been shown to improve robustness by introducing auxiliary supervision at intermediate layers [16]. To leverage LID supervision, previous work like [18] applies an auxiliary LID CTC loss at the *downstream* intermediate encoder layers, conditioning later layers on LID predictions. Another approach extracts language-related frames by average pooling to guide language prediction [23]. Unlike these approaches, we incorporate LID CTC loss directly into *upstream* fine-tuning, enabling language-aware representation learning earlier in the upstream.

3. Proposed method

3.1. Training strategies with multilingual SFMs

To evaluate the performance of the latest multilingual SFMs on the ML-SUPERB 2.0 benchmark, we select three representa-

10.21437/Interspeech.2025-2377

tive encoder-only SFMs as the upstream models: the SSL-based MMS [5] and XEUS [6], and supervised pre-trained OWSM-CTC [11]. The overall architecture, as shown in Figure 1, consists of an SFM as the upstream encoder, followed by a downstream model trained with the CTC loss. Three training strategies are adopted: training downstream with upstream frozen, upstream model fine-tuning, and parameter-efficient upstream model adaptation.

3.1.1. *Upstream models*

MMS and XEUS share a common architecture, both starting with a convolutional waveform encoder that extracts a T-length sequence of D-dimensional acoustic features $X \in \mathbb{R}^{T \times D}$ from raw audio. These features are then processed by a stack of L encoder layers ($\{f^l_{\text{ENC}}\}_{l=1}^{L}$), using Transformer [24] in MMS and E-Branchformer [25] in XEUS:

$$Z^l = f^l_{\text{ENC}}(Z^{l-1}), \tag{1}$$

where $Z^l \in \mathbb{R}^{T \times D}$ is the l-th layer output, with $Z^0 = X$.

Unlike MMS and XEUS, which directly process raw waveforms, OWSM-CTC [11] takes log Mel filter banks as input, which are processed by a convolutional encoder followed by L stacked E-Branchformer [25] encoder layers ($\{f^l_{\text{ENC}}\}_{l=1}^{L}$). OWSM-CTC employs an encoder-only architecture with a prompt encoder. For fair comparison in ML-SUPERB 2.0, we evaluate only its speech encoder with speech input, excluding prefix tokens and other components.

3.1.2. *Training downstream models with frozen upstream*

To assess the cross-lingual speech representation ability of SSL and supervised model encoders, we use them as upstream models and stack Transformer encoder layers as downstream models. These downstream layers further model contextual dependencies given the upstream representations. During training, the upstream parameters are frozen and only the downstream layers are optimized. To aggregate upstream representations, we apply a weighted sum approach [26, 27] to derive the final output:

$$Z_{\text{out}} = \sum_{l=1}^{L} \alpha_l \cdot Z^l, \tag{2}$$

where α_l is a learnable parameter satisfying $\sum_{l=1}^{L} \alpha_l = 1$.

To adapt upstream representations, $Z_{\text{out}} \in \mathbb{R}^{T \times D}$ is linearly projected to a lower feature dimension of D_{proj} and then mapped to the downstream hidden dimension D_{down} through an input layer using either a subsampling convolution (with factor κ) or a linear layer (i.e. κ = 1). Positional encoding is then added before feeding the transformed features into the subsequent downstream encoder layers, producing the final output $H_{\text{out}} \in \mathbb{R}^{T/\kappa \times D_{\text{down}}}$.

Given the S-length target sequence $Y = (y_s \in \mathcal{V} | s = 1, \ldots, S)$, where $\mathcal{V}$ is a vocabulary consisting of linguistic tokens and language codes, the CTC loss is defined as:

$$P_{\text{CTC}}(Y|H_{\text{out}}) = \sum_{\pi \in \mathcal{F}^{-1}(Y)} \prod_{t=1}^{T/\kappa} P(\pi_t|H_{\text{out}}), \tag{3}$$

$$\mathcal{L}_{\text{CTC}} = -\log P_{\text{CTC}}(Y|H_{\text{out}}), \tag{4}$$

where $\mathcal{F}^{-1}(Y)$ denotes the set of all latent alignments between the downstream output H_{out} and target sequence Y, and π represents a specific alignment. During training, we freeze the upstream and fully optimize the downstream using the CTC loss.

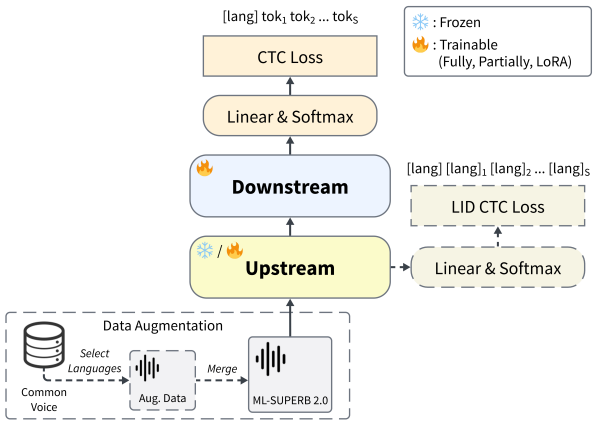

Figure 1: *Model architecture. The upstream model is frozen, partially fine-tuned, or adapted with LoRA. The downstream model is fully trainable. Training uses CTC loss, with optional data augmentation and LID CTC loss. Dashed lines denote optional enhancements; solid lines represent core processes.*

3.1.3. *Fine-tuning upstream models*

While training only the downstream layers leverages pre-trained representations from the upstream model, it may limit the upstream model's adaptability to target languages. To address this, we fine-tune the upstream model by unfreezing partial layers rather than all layers, as full-layer fine-tuning is computationally expensive. Specifically, we optimize the bottom, middle, or top consecutive layers $\mathcal{N} \subset \{1, \ldots, L\}$, with the number and position of unfrozen layers varying across different upstream models. During fine-tuning, we use the same downstream architectures as in Section 3.1.2, but with shallower encoder layers, optimized with the CTC loss to better adapt upstream representations for multilingual LID and ASR.

3.1.4. *Parameter-efficient upstream model adaptation*

Full-parameter fine-tuning is computationally expensive due to numerous trainable parameters. To optimize efficiently, we use LoRA [28], which freezes pre-trained parameters and injects trainable low-rank matrices into self-attention layers. We apply LoRA to all self-attention projection weights at each upstream layer. As before, shallow encoder layers are used in the downstream model, with the full model optimized with the CTC loss.

3.2. Data augmentation for low-resource languages

The ML-SUPERB 2.0 official training data includes 20 few-shot languages, each with at most 5 utterances in the training set [12]. Due to limited resources, both LID and ASR performance degrade from insufficient language-specific supervision. To address this, we augment the ML-SUPERB 2.0 official training set with additional data for underperforming languages, as illustrated in Figure 1. Specifically, we select data from the Common Voice Corpus [29] (version 20.0), covering some few-shot and underperforming languages present in the baseline results.[1] For each language, we randomly sample an average of 2123 utterances, with exact counts varying. To mitigate the language bias, the sampled data is integrated into

[1]ISO 639-3 codes of augmented languages: vie, srp, lit, dan, frr, tur, kaz, zul, epo, tok, ory, ast, lao, nan, cmn, kat, kor, jpn, yue, lit, kab, heb.

the ML-SUPERB 2.0 official training data, and the models are trained with the augmented training set.

3.3. Auxiliary LID CTC loss

ML-SUPERB 2.0 provides an average of 1.6 hours of training data per language, which can lead to overfitting during fine-tuning. To regularize fine-tuning and improve model generalizability, we introduce an auxiliary LID prediction task with an LID CTC objective, as depicted in Figure 1. Specifically, we define a language code sequence $I = (i_s \in \mathcal{I} \mid s = 1, \ldots, S)$, where $\mathcal{I}$ is the set of language codes corresponding to languages in the training set. Each i_s in I is identical, representing the same language for the entire utterance. The sequence length S matches that of the transcription sequence Y in Equation 3. Given the output Z^l of the l-th upstream fine-tuning encoder layer f^l_{ENC}, the LID CTC loss for layer l is defined as:

$$P^l_{\text{CTC}}(I|Z^l) = \sum_{\varphi \in \mathcal{G}^{-1}(I)} \prod_{t=1}^{T} P(\varphi_t|Z^l), \tag{5}$$

$$\mathcal{L}^l_{\text{LID}} = -\log P^l_{\text{CTC}}(I|Z^l), \tag{6}$$

where $\mathcal{G}^{-1}(I)$ represents the set of possible alignments between Z^l and I, and φ denotes a specific alignment. The LID CTC loss is applied to a subset $\mathcal{M}$ of fine-tuning layers, where $\mathcal{M} \subseteq \mathcal{N}$ and $\mathcal{N}$ denotes the set of fine-tuning layers mentioned in Section 3.1.3. The final training objective combines the primary ASR CTC loss with the auxiliary LID CTC loss:

$$\mathcal{L} = (1-\beta)\mathcal{L}_{\text{CTC}} + \beta \frac{\sum_{l \in \mathcal{M}} \mathcal{L}^l_{\text{LID}}}{|\mathcal{M}|}, \quad \text{where } \beta \in [0, 1]. \tag{7}$$

4. Experiments

General setup. We use the ML-SUPERB 2.0 official dataset from the ML-SUPERB Challenge 2025, which includes a training set (220 hours, 138 languages), standard development set (41 hours, 138 languages), and dialect development set (9 hours, 56 dialects).[2] Models are evaluated on the development sets due to the unavailability of the test set. We adopt the baseline provided by the challenge, using frozen MMS with 1 B parameters (MMS-1B) as the frozen upstream model, with its layer outputs weighted summed and fed into a two-layer Transformer encoder.[3] We re-run this baseline model in our evaluation framework to ensure fair comparison, rather than using pre-reported results. We also report results based on XEUS [6] and XLS-R 128 (300 M) [30] provided by the challenge, although the implementation details are not disclosed. The largest feasible SFMs or newest variants are used as our upstream models: MMS-1B[4] (1 B parameters), XEUS[5] (577 M), and OWSM-CTC v4 medium (1.01 B) [31]. A character tokenizer is used for all models. All experiments are conducted using ESPnet [32], with S3PRL [27] serving as the upstream model interface.

Model configuration. Each model has projection dimension D_{proj} = 80 (Section 3.1.2). The input layer uses a subsampling convolutional neural network with factor κ = 2 and kernel size 3, reducing the frame rate to 40 ms for MMS and XEUS, while OWSM-CTC uses a linear layer to retain 80 ms. The downstream models employ Transformer encoders with sinusoidal positional encodings [24]. To ensure fair comparisons, configurations for frozen upstream training and upstream fine-tuning are designed to maintain a consistent trainable parameter size (around 260 M). The downstream model for training with frozen upstream consists of 12 layers, 1320 hidden units (D_{down} in Section 3.1.2), 8 heads, and 5280 projection units. For upstream fine-tuning, it consists of 2 layers, 512 hidden units (D_{down}), 8 heads, and 2048 projection units. The specific fine-tuned upstream layers, selected from the bottom, middle, or top layers, are listed in Table 1, with the layer count adjusted to keep trainable parameters consistent. The LoRA-based strategy follows the same setting but with 1 layer downstream encoder, and its rank and scaling factor are set to 16.

Training setup. All models are trained for 10 epochs, with a batch size of 8 and 30000 steps in each epoch. Gradient accumulation is applied every 4 steps to stabilize updates. The learning rate is tuned over 1×10^{-5}, 3×10^{-5}, 1×10^{-4} and 3×10^{-4} with 1×10^{-4} chosen, optimized with Adam optimizer [33]. A 25000-step warm-up is used for training downstream with frozen upstream and LoRA except for upstream fine-tuning. SpecAugment [22] is applied to the upstream output. The final model is selected based on the lowest validation loss.

Fine-tuning with augmented data and LID CTC. Based on Table 1, we select the optimal fine-tuning layer set $\mathcal{N}$ for each upstream model: $\{25, \ldots, 36\}$ for MMS-1B, $\{12, \ldots, 19\}$ for XEUS, and $\{8, \ldots, 13\}$ for OWSM-CTC. These configurations are then used for training on augmented data (Section 3.2). For the LID CTC loss (Section 3.3), the applied layer subset $\mathcal{M}$ is $\{27, 30, 33, 36\}$ for MMS-1B, $\{14, 17\}$ for XEUS, and $\{10, 13\}$ for OWSM-CTC. We evaluate it on both the official and augmented ML-SUPERB 2.0 training sets, tuning β in Equation 7 over 3×10^{-2}, 1×10^{-1} and 3×10^{-1} with 3×10^{-1} chosen.

5. Results

5.1. Training strategies with multilingual SFMs

Table 1 shows LID and ASR results across SFMs and training strategies on ML-SUPERB 2.0. On the standard set, MMS-1B with LoRA achieves the best performance (82.4% LID accuracy, 16.3% CER). On the dialect set, fine-tuning the middle layers of MMS-1B achieves the highest LID accuracy (68.4%), while LoRA on MMS-1B yields the lowest CER (28.5%). XEUS performs competitively but lags behind MMS-1B, likely due to its smaller size (about 60% of MMS-1B). Notably, our best strategies with MMS-1B and XEUS outperform all baselines on the standard set but trail the XEUS baseline on the dialect set. Additionally, our methods exhibit lower overall deviations, suggesting improved robustness across languages. OWSM-CTC performs the worst, especially in ASR, as its pre-training lacks full ML-SUPERB 2.0 language coverage.

Which SFM is more effective for downstream? In our setting, XEUS outperforms MMS-1B and OWSM-CTC when training with frozen upstream representations. Compared to the MMS-1B (Baseline), MMS-1B (T.) with a larger downstream model degrades performance, likely due to overfitting from increased depth and hidden size. OWSM-CTC (T.) underperforms the most, likely due to the omission of the prefix token input [11] and its restricted language coverage.

Fine-tune which layers is more effective? For MMS-1B, fine-tuning middle layers (13-24, 25-36) outperforming bottom (1-12) and top layers (37-48), especially in dialect LID and standard ASR. However, XEUS benefits more from fine-tuning upper layers on the standard set, achieving its best fine-tuning performance on 12-19 layers, while for dialect LID and ASR, mid-

[2] https://huggingface.co/datasets/espnet/ml_superb_hf
[3] https://huggingface.co/espnet/mms_1b_mlsuperb
[4] https://huggingface.co/facebook/mms-1b
[5] https://huggingface.co/espnet/xeus

Table 1: *Results of different models and strategies on ML-SUPERB 2.0. LID accuracy (ACC, %) and ASR CER (%) are averaged across languages for full and worst-15 (WL) in the standard set, and dialects (DL). An asterisk (*) on Baseline denotes an unknown training strategy. T. indicates Transformer downstream. Numbers (e.g., 1-12) represent fine-tuned layers. Best per model is underlined, overall best is **bolded**.*

Model	Method	LID (ACC ↑)		ASR (CER ↓)		
		Full	DL	Full	WL	DL
MMS-1B	Baseline	76.1	58.4	22.3 ± 14.5	55.4	32.7
XEUS	Baseline*	77.1	79.1	22.4 ± 30.9	78.9	23.2
XLS-R 128	Baseline*	72.4	60.7	31.7 ± 26.9	86.8	30.5
MMS-1B	T.	69.6	50.6	27.9 ± 18.8	72.6	41.7
	1-12	80.7	63.9	20.5 ± 14.7	54.5	38.2
	13-24	81.5	68.4	18.9 ± 13.7	49.3	36.8
	25-36	80.2	**68.4**	17.1 ± 13.0	46.6	34.5
	37-48	78.2	52.3	20.7 ± 15.5	55.3	43.9
	LoRA	**82.4**	64.9	**16.3 ± 12.7**	**44.5**	**28.5**
XEUS	T.	79.2	61.5	23.8 ± 12.8	50.2	39.9
	1-8	74.7	52.7	25.6 ± 14.7	57.5	47.6
	7-14	77.0	67.9	23.6 ± 14.7	57.0	38.6
	12-19	77.3	56.9	21.6 ± 14.8	54.9	43.1
	LoRA	76.6	48.8	26.5 ± 18.2	69.4	43.9
OWSM-CTC	T.	57.9	38.4	45.8 ± 18.3	83.2	66.4
	1-6	79.0	66.7	30.2 ± 15.7	63.6	50.1
	8-13	78.3	67.7	28.2 ± 13.5	56.4	49.9
	15-20	74.3	62.1	29.5 ± 13.5	57.7	52.7
	22-27	63.4	52.5	37.1 ± 19.5	80.8	60.6
	LoRA	74.3	64.3	32.4 ± 13.0	59.7	52.8

level layers (7-14) perform best. Similar to MMS-1B, OWSM-CTC achieves lower CER by fine-tuning middle layers (8-13, 15-20) than bottom or top layers. These results indicate that different models exhibit varying layer-wise encoding properties, requiring model-specific fine-tuning strategies.

LoRA benefits MMS-1B more. The Transformer-based MMS-1B with LoRA achieves the best overall performance over fine-tuning and frozen upstream training. However, LoRA for E-Branchformer-based XEUS and OWSM-CTC fails to surpass fine-tuning. This may be because our LoRA configuration adapts only self-attention projections, which are central to Transformer. In contrast, E-Branchformer's hybrid design with a local extractor [25] may require broader adaptation, limiting LoRA's effectiveness.

5.2. Fine-tuning with data augmentation and LID CTC

Data augmentation boosts LID and ASR on few-shot languages. Table 2 presents the LID and ASR results for fine-tuning with data augmentation and LID CTC, explicitly considering performance on few-shot languages. With data augmentation, the most significant improvements are observed in LID and ASR for few-shot languages, highlighting the benefit of additional training data for low-resource languages. For instance, fine-tuning XEUS (12-19) on augmented data improves few-shot language LID accuracy by 42.7% and reduces CER by 17.2% absolutely compared to using the official dataset. Similarly, MMS-1B and OWSM-CTC see notable performance improvements on few-shot languages, confirming the benefits of augmentation across various architectures. This improvement

Table 2: *Results on fine-tuning with data augmentation and LID CTC. Metrics follow Table 1, with results for few-shot (FS) languages in the standard set (Std.) explicitly reported. Numbers indicate fine-tuned layers. LIDCTC: fine-tuning with LID CTC on official data. DataAug: training on augmented data. DataAug*: fine-tuning with LID CTC on augmented data.*

Model	Method	LID (ACC ↑)			ASR (CER ↓)			
		Std.		DL	Std.			DL
		Full	FS		Full	FS	WL	
MMS-1B	25-36	80.2	2.4	68.4	17.1 ± 13.0	24.6	46.6	34.5
	LIDCTC	82.9	6.1	66.2	17.1 ± 13.5	27.9	47.7	30.3
	DataAug	86.0	40.0	63.9	15.9 ± 10.7	15.2	39.3	34.3
	DataAug*	86.9	40.7	74.2	15.6 ± 10.5	15.1	38.6	31.5
XEUS	12-19	77.3	0.0	56.9	21.6 ± 14.8	33.1	54.9	43.1
	LIDCTC	79.9	4.4	68.7	21.4 ± 14.6	33.1	55.4	34.8
	DataAug	84.8	42.7	55.5	18.1 ± 10.5	15.9	41.5	39.4
	DataAug*	85.8	44.6	69.5	17.9 ± 10.5	16.4	42.0	35.4
OWSM-CTC	8-13	78.3	4.7	67.7	28.2 ± 13.5	33.8	56.4	49.9
	LIDCTC	76.4	2.2	62.4	29.3 ± 13.2	33.3	57.0	52.2
	DataAug	81.4	42.3	64.7	27.6 ± 11.7	21.7	50.5	52.4
	DataAug*	82.0	39.9	64.5	27.5 ± 11.5	21.7	50.3	50.8

suggests that data augmentation enhances model robustness by increasing data diversity and mitigating overfitting.

LID CTC enhances dialect performance for MMS-1B and XEUS. Beyond data augmentation, applying LID CTC during fine-tuning introduces additional improvements for MMS-1B and XEUS, especially for dialects. For XEUS fine-tuned on the official training set, dialect LID accuracy increases by 11.8% and ASR CER decreases by 8.3% absolutely. While MMS-1B sees a slight drop in dialect LID accuracy on the official dataset, it benefits significantly from LID CTC when trained on augmented data, improving dialect LID accuracy by 10.3% and reducing dialect ASR CER by 2.8% absolutely. This suggests that auxiliary LID prediction enhances generalization, though sufficient data may be required. Unlike MMS-1B and XEUS, applying LID CTC loss to OWSM-CTC leads to slight degradation with both official and augmented data. This suggests that the auxiliary LID CTC loss may interfere with OWSM-CTC's supervised pre-training, which is already optimized for LID and ASR, limiting further improvements.

6. Conclusion

In this paper, we explore strategies to improve multilingual LID and ASR performance on ML-SUPERB 2.0. We evaluate MMS, XEUS, and OWSM-CTC under downstream training with frozen upstream, upstream fine-tuning, and LoRA. Furthermore, we incorporate data augmentation and LID CTC loss for fine-tuning. Results show that MMS-1B outperforms XEUS and OWSM-CTC under our strategies, especially with LoRA and mid-layer fine-tuning. Data augmentation improves few-shot performance, while LID CTC loss enhances generalization on dialects. Overall, our MMS-1B-based model, fine-tuned with LID CTC loss on augmented data, achieves **second place** in the Interspeech 2025 ML-SUPERB 2.0 Challenge. Future work will explore language model integration and strategies to further improve low-resource language performance.

7. Acknowledgements

Experiments of this work used the Bridges2 system at PSC and Delta system at NCSA through allocations CIS210014 and IRI120008P from the Advanced Cyberinfrastructure Coordination Ecosystem: Services & Support (ACCESS) program, supported by National Science Foundation grants #2138259, #2138286, #2138307, #2137603, and #2138296.

8. References

[1] J. S. Peter K. Austin, Ed., *The Cambridge Handbook of Endangered Languages* (Cambridge Handbooks in Language and Linguistics). Cambridge University Press, 2011.

[2] A. Babu, C. Wang, A. Tjandra, K. Lakhotia, *et al.*, "XLS-R: Self-supervised Cross-lingual Speech Representation Learning at Scale," in *Proc. Interspeech*, 2021, pp. 2278–2282.

[3] X. Li, F. Metze, D. R. Mortensen, A. W. Black, *et al.*, "ASR2K: Speech recognition for around 2000 languages without audio," in *Proc. Interspeech*, 2022, pp. 4885–4889.

[4] W. Chen, J. Shi, B. Yan, D. Berrebbi, *et al.*, "Joint prediction and denoising for large-scale multilingual self-supervised learning," in *Proc. ASRU*, 2023, pp. 1–8.

[5] V. Pratap, A. Tjandra, B. Shi, P. Tomasello, *et al.*, "Scaling speech technology to 1,000+ languages," *Journal of Machine Learning Research*, vol. 25, no. 97, pp. 1–52, 2024.

[6] W. Chen, W. Zhang, Y. Peng, X. Li, *et al.*, "Towards robust speech representation learning for thousands of languages," in *Proc. EMNLP*, 2024, pp. 10 205–10 224.

[7] M. Zanon Boito, V. Iyer, N. Lagos, L. Besacier, *et al.*, "mHuBERT-147: A compact multilingual HuBERT model," in *Proc. Interspeech*, 2024, pp. 3939–3943.

[8] L. Zhou, J. Li, E. Sun, and S. Liu, "A configurable multilingual model is all you need to recognize all languages," in *Proc. ICASSP*, 2021, pp. 6422–6426.

[9] A. Radford, J. W. Kim, T. Xu, G. Brockman, *et al.*, "Robust speech recognition via large-scale weak supervision," in *Proc. ICML*, 2023, pp. 28 492–28 518.

[10] Y. Peng, J. Tian, W. Chen, S. Arora, *et al.*, "OWSM v3.1: Better and faster open Whisper-style speech models based on E-Branchformer," in *Proc. Interspeech*, 2024, pp. 352–356.

[11] Y. Peng, Y. Sudo, M. Shakeel, and S. Watanabe, "OWSM-CTC: An open encoder-only speech foundation model for speech recognition, translation, and language identification," in *Proc. ACL*, 2024, pp. 10 192–10 209.

[12] J. Shi, S.-H. Wang, W. Chen, M. Bartelds, *et al.*, "ML-SUPERB 2.0: Benchmarking multilingual speech models across modeling constraints, languages, and datasets," in *Proc. Interspeech*, 2024, pp. 1230–1234.

[13] J. Shi, D. Berrebbi, W. Chen, H.-L. Chung, *et al.*, "ML-SUPERB: Multilingual speech universal performance benchmark," in *Proc. Interspeech*, 2023, pp. 884–888.

[14] A. Graves, S. Fernández, F. Gomez, and J. Schmidhuber, "Connectionist temporal classification: Labelling unsegmented sequence data with recurrent neural networks," in *Proc. ICML*, 2006, pp. 369–376.

[15] A. Tjandra, C. Liu, F. Zhang, X. Zhang, *et al.*, "Deja-vu: Double feature presentation and iterated loss in deep transformer networks," in *Proc. ICASSP*, 2020, pp. 6899–6903.

[16] J. Lee and S. Watanabe, "Intermediate loss regularization for CTC-based speech recognition," in *Proc. ICASSP*, 2021, pp. 6224–6228.

[17] J. Nozaki and T. Komatsu, "Relaxing the conditional independence assumption of CTC-based ASR by conditioning on intermediate predictions," in *Proc. Interspeech*, 2021, pp. 3735–3739.

[18] W. Chen, B. Yan, J. Shi, Y. Peng, *et al.*, "Improving massively multilingual ASR with auxiliary CTC objectives," in *Proc. ICASSP*, 2023, pp. 1–5.

[19] B. Li, R. Pang, Y. Zhang, T. N. Sainath, *et al.*, "Massively multilingual ASR: A lifelong learning solution," in *Proc. ICASSP*, 2022, pp. 6397–6401.

[20] M. Bartelds, N. San, B. McDonnell, D. Jurafsky, *et al.*, "Making more of little data: Improving low-resource automatic speech recognition using data augmentation," in *Proc. ACL*, 2023, pp. 715–729.

[21] T. Ko, V. Peddinti, D. Povey, and S. Khudanpur, "Audio augmentation for speech recognition," in *Proc. Interspeech*, 2015, pp. 3586–3589.

[22] D. S. Park, W. Chan, Y. Zhang, C.-C. Chiu, *et al.*, "Specaugment: A simple data augmentation method for automatic speech recognition," in *Proc. Interspeech*, 2019, pp. 2613–2617.

[23] H. Xue, Q. Shao, K. Huang, P. Chen, *et al.*, "SSHR: Leveraging self-supervised hierarchical representations for multilingual automatic speech recognition," in *Proc. ICME*, 2024, pp. 1–6.

[24] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, *et al.*, "Attention is all you need," in *Proc. NeurIPS*, 2017, pp. 6000–6010.

[25] K. Kim, F. Wu, Y. Peng, J. Pan, *et al.*, "E-Branchformer: Branchformer with enhanced merging for speech recognition," in *Proc. SLT*, 2023, pp. 84–91.

[26] M. E. Peters, M. Neumann, M. Iyyer, M. Gardner, *et al.*, "Deep contextualized word representations," in *Proc. NAACL*, 2018, pp. 2227–2237.

[27] S.-w. Yang, P.-H. Chi, Y.-S. Chuang, C.-I. J. Lai, *et al.*, "SUPERB: Speech processing universal performance benchmark," in *Proc. Interspeech*, 2021, pp. 1194–1198.

[28] E. J. Hu, yelong shen, P. Wallis, Z. Allen-Zhu, *et al.*, "LoRA: Low-rank adaptation of large language models," in *Proc. ICLR*, 2022.

[29] R. Ardila, M. Branson, K. Davis, M. Kohler, *et al.*, "Common voice: A massively-multilingual speech corpus," in *Proc. LREC*, 2020, pp. 4218–4222.

[30] A. Babu, C. Wang, A. Tjandra, K. Lakhotia, *et al.*, "XLS-R: Self-supervised cross-lingual speech representation learning at scale," in *Proc. Interspeech*, 2022, pp. 2278–2282.

[31] Y. Peng, S. Muhammad, Y. Sudo, W. Chen, *et al.*, "OWSM v4: Improving open Whisper-style speech models via data scaling and cleaning," in *Proc. Interspeech*, 2025.

[32] S. Watanabe, T. Hori, S. Karita, T. Hayashi, *et al.*, "ESPnet: End-to-end speech processing toolkit," in *Proc. Interspeech*, 2018, pp. 2207–2211.

[33] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *Proc. ICLR (Poster)*, 2015.