MITSUBISHI ELECTRIC RESEARCH LABORATORIES
https://www.merl.com

Momentum Pseudo-Labeling: Semi-Supervised ASR with Continuously Improving Pseudo-Labels

Higuchi, Yosuke; Moritz, Niko; Le Roux, Jonathan; Hori, Takaaki

TR2022-112 September 29, 2022

Abstract

Safety and robustness are two desired properties for any reinforcement learning algorithm. Constrained Markov Decision Processes (CMDPs) can handle additional safety constraints and Robust Markov Decision Processes (RMDPs) can perform well under model uncertainties. In this chapter, we propose to unify these two frameworks resulting in Robust Constrained MDPs (RCMDPs). The motivation is to develop a framework that can satisfy safety constraints while also simultaneously offer robustness to model uncertainties. We develop the RCMDP objective, derive gradient update formula to optimize this objective and then propose policy gradient based algorithms. We also independently propose Lyapunov-based reward shaping for RCMDPs, yielding better stability and convergence properties.

IEEE Journal of Selected Topics in Signal Processing 2022

Mitsubishi Electric Research Laboratories, Inc.
201 Broadway, Cambridge, Massachusetts 02139

Momentum Pseudo-Labeling: Semi-Supervised ASR with Continuously Improving Pseudo-Labels

Yosuke Higuchi, *Student Member, IEEE,* Niko Moritz, *Member, IEEE,*
Jonathan Le Roux, *Senior Member, IEEE* and Takaaki Hori, *Senior Member, IEEE*

***Abstract*—End-to-end automatic speech recognition (ASR) has become a popular alternative to traditional module-based systems, simplifying the model-building process with a single deep neural network architecture. However, the training of end-to-end ASR systems is generally data-hungry: a large amount of labeled data (speech-text pairs) is necessary to learn direct speech-to-text conversion effectively. To make the training less dependent on labeled data, pseudo-labeling, a semi-supervised learning approach, has been successfully introduced to end-to-end ASR, where a seed model is self-trained with pseudo-labels generated from unlabeled (speech-only) data. Here, we propose *momentum pseudo-labeling* (MPL), a simple yet effective strategy for semi-supervised ASR. MPL consists of a pair of *online* and *offline* models that interact and learn from each other, inspired by the mean teacher method. The online model is trained to predict pseudo-labels generated on the fly by the offline model. The offline model maintains an exponential moving average of the online model parameters. The interaction between the two models allows better ASR training on unlabeled data by continuously improving the quality of pseudo-labels. We apply MPL to a connectionist temporal classification-based model and evaluate it on various semi-supervised scenarios with varying amounts of data or domain mismatch. The results demonstrate that MPL significantly improves the seed model by stabilizing the training on unlabeled data. Moreover, we present additional techniques, e.g., the use of Conformer and an external language model, to further enhance MPL, which leads to better performance than other semi-supervised methods based on pseudo-labeling.**

***Index Terms*—pseudo-labeling, self-training, semi-supervised learning, end-to-end speech recognition, deep learning.**

I. Introduction

THE field of automatic speech recognition (ASR) has witnessed remarkable improvements in performance thanks to the advances in deep learning-based techniques [1], [2]. Much of the recent progress in ASR lies in the end-to-end framework [3]–[5], which directly models speech-to-text conversion using a single deep neural network. With well-established sequence-to-sequence modeling techniques [6]–[9] and sophisticated neural network architectures [10]–[12], end-to-end ASR has demonstrated promising results on various benchmarks [13]–[15]. However, the performance often depends on the availability of a large quantity of labeled data (speech-text pairs) [16], which requires great annotation costs and is not always achievable.

To alleviate the heavy requirement for labeled data, semi-supervised learning [17] has been attracting increasing attention for improving end-to-end ASR. Semi-supervised learning utilizes labeled data as well as unlabeled (or unpaired) data during model training, where the amount of labeled data is in general much smaller than that of unlabeled data. Some early works for semi-supervised end-to-end ASR are based on a reconstruction framework, including approaches based on a text-to-speech model [18]–[20] or a sequential auto-encoder [21]–[23]. Others adopted self-supervised pre-training techniques, such as BERT-like mask prediction [24]–[26], contrastive learning [27]–[29], and feature clustering [30], [31], to boost the performance of downstream ASR tasks.

We focus on self-training [32] or *pseudo-labeling* [33], which has recently been adopted for semi-supervised end-to-end ASR and shown to be effective [34]–[44]. In pseudo-labeling, a teacher model is first trained on labeled data and used to transcribe unlabeled (speech-only) data to obtain pseudo-labels. A student model is then trained using both the labeled and pseudo-labeled data to achieve better performance than the teacher. Assuming external text data is accessible, external language models (LMs) and beam-search decoding are often incorporated into the labeling process to generate higher-quality pseudo-labels [35], [38]. Data augmentation is also important for assisting a student model with training on pseudo-labels [36], [37], [40]. In addition to these techniques, ASR performance can be further improved by iterating the pseudo-labeling steps [39]–[43]. In [40], a model is continuously trained on pseudo-labels, which are generated on the fly by the model itself. Pseudo-labels are refined as the model learns, and the model benefits from training on the refined labels. However, we observed that this frequent update of pseudo-labels can easily cause unstable training, especially when there is a large amount of unlabeled data or domain mismatch between labeled and unlabeled data, which is likely to be the case in real-world scenarios.

In this paper, we present a semi-supervised learning framework for end-to-end ASR, referred to as *momentum pseudo-labeling* (MPL). In MPL, the pseudo-labels are iteratively updated based on an ensemble of models at different time steps within a single training process [45]. MPL consists of *online* and *offline* models that interact and learn from each other, similar to the teacher-student framework in the mean teacher

Manuscript received January 15, 2022; revised June 18, 2022; revised July 15, 2022. *(Corresponding author: Yosuke Higuchi.)*

Y. Higuchi is with Waseda University, Tokyo 162-0042, Japan (e-mail: higuchi@pcl.cs.waseda.ac.jp).

N. Moritz was with Mitsubishi Electric Research Laboratories (MERL), Cambridge, MA 02139, USA, and is now with Meta AI, London W1T 1FB, UK (e-mail: nmoritz@fb.com).

J. Le Roux is with Mitsubishi Electric Research Laboratories (MERL), Cambridge, MA 02139, USA (e-mail: leroux@merl.com).

T. Hori was with Mitsubishi Electric Research Laboratories (MERL), Cambridge, MA 02139, USA, and is now with Apple, Inc., Cambridge, MA 02142, USA (e-mail: thori@ieee.com).

method [46]. The online model is trained to predict pseudo-labels generated on the fly by the offline model. The offline model maintains an exponential moving average of the weights of the online model, which can be regarded as an ensemble of the online models at different training steps. Through the interaction between the two models, MPL effectively stabilizes the training with unlabeled data and handles the constant change in pseudo-labels.

The contributions of this paper are summarized as follows:

- We propose MPL and show its advantages over other semi-supervised approaches based on pseudo-labeling.
- We present an effective way for controlling the MPL training, which reduces the burden for heuristic tuning.
- We evaluate MPL in various semi-supervised scenarios, which demonstrates its robustness against variations in the amount of data, variations in domain mismatch severity, and over-fitting to LM knowledge.
- We perform thorough analyses to confirm the effectiveness of MPL and propose several methods to further improve ASR performance.

This paper summarizes our previous studies on MPL [47], [48] with the following extensions: we provide more detailed explanations of relationship to prior works (Section II) and precise formulations of end-to-end ASR and pseudo-labeling (Section III); we present a consistent description of [47] and [48], with more specific implementations (Section IV); we conduct experiments on a variety of semi-supervised scenarios, including additional experiments on smaller and larger amounts of labeled data (Section V); and we further demonstrate the effectiveness of MPL through more detailed experimental results and discussions (Section VI).

II. Related Work

A. Self-ensembling for semi-supervised learning

Self-ensembling [45] is a semi-supervised learning framework, where a target of an unlabeled sample is obtained by a consensus of predictions from models at different training steps or different models. This prediction ensembling is expected to produce a more accurate pseudo-label than the most recent model prediction. Several approaches have been proposed to implement self-ensembling; we here refer to techniques based on an exponential moving average (EMA). Temporal ensembling [45] maintains an EMA of label predictions from different models, which is used as a target for model training at the current step. The mean teacher method [46] improves temporal ensembling by calculating an EMA of model weights and generating a pseudo-label using the averaged model. This can avoid sudden changes in pseudo-labels and enable the model to learn from unlabeled samples stably. The concept of EMA-based weight averaging has also been shown to be effective for stabilizing self-supervised representation learning [49], [50].

MPL is inspired by and similar to the mean teacher framework. However, we differentiate MPL from prior work in the following perspectives. 1) MPL is a semi-supervised learning framework for end-to-end ASR: while most previous studies focus on classification problems (e.g., image classification), few have introduced self-ensembling techniques to sequence-to-sequence mapping objectives, here connectionist temporal classification (CTC) [6]. 2) MPL uses hard (pseudo-)labels for training with unlabeled data: while soft labels generally contain richer information for promoting a model training [51], applying a distillation loss to CTC-based ASR systems is known to be problematic [52]; as CTC models emit spiky posterior distributions and predictions are naturally high-confidence, we consider hard labels more suitable for MPL.[1] 3) MPL applies data augmentation (i.e., SpecAugment [55]) to the input only for training the online model, while the offline model generates pseudo-labels in inference mode: since we do not use soft labels in MPL, it is preferable for pseudo-labels to be accurate; moreover, the online model can learn to robustly predict pseudo-labels from noisy input, an effective approach known as consistency training [36], [40], [56].

B. Pseudo-labeling with multiple iterations

A simple extension for enhancing the pseudo-labeling-based method is to conduct multiple rounds of the pseudo-label generation and model training processes, demonstrating promising results in various fields [57], [58] including end-to-end ASR [39]–[43]. Iterative pseudo-labeling (IPL) [39], an iterative version of [35], continuously trains a single ASR model with periodic regeneration of pseudo-labels. Here, the labeling process is performed via beam-search decoding with an external LM, which makes the pseudo-labels biased toward LM training texts and the model over-fit to the LM knowledge [42], [43]. In [40], an ASR model is trained on pseudo-labels generated without using an LM, where the pseudo-labels are updated on the fly after every training iteration. However, this frequent relabeling is likely to make pseudo-labels unstable and thus cause the model training to diverge. slimIPL [42] mitigates this problem by introducing a dynamic cache mechanism, which stores and uses pseudo-labels generated from the previous model states instead of regenerating them with the most recent model every iteration.

MPL is another direction for improving pseudo-labeling with multiple iterations, which can be considered as a general framework extending [35] and [40] (see Section IV-B). In each iteration of MPL training, pseudo-labels are generated on the fly from the offline model without an LM and used as targets to train the online model. The offline model maintains an EMA of the online model weights to stabilize pseudo-labels. This can be seen as alternative caching mechanism to [42] for exploiting older models. A similar approach to MPL was proposed in [59], which focused on lower-resource settings and conducted experiments on a hybrid ASR system in addition to a CTC-based end-to-end system. This paper thoroughly investigates MPL on its robustness against variations in domain mismatch severity and over-fitting to LM knowledge.

[1] Several works have proposed practical approaches for distilling frame-level knowledge between CTC-based models [53], [54]. As in TutorNet [54], we tried MPL training with the l_2 loss between the online and offline models, but we did not observe a significant improvement. Hence in this work, we only focus on using hard (pseudo-)labels for simplicity.

III. Background

In this section, we review a formulation of CTC-based end-to-end ASR [3] and semi-supervised ASR based on pseudo-labeling [35]. Let $X = (\mathbf{x}_t \in \mathbb{R}^D | t = 1, \dots, T')$ be an input sequence of length T', and $Y = (y_l \in \mathcal{V} | l = 1, \dots, L)$ be the corresponding output sequence of length L. Here, $\mathbf{x}_t$ is a D-dimensional acoustic feature at frame t, y_l is an output token at position l, and $\mathcal{V}$ is a vocabulary. Note that, in general, the output length is much shorter than the input length (i.e., $L \ll T'$).

A. Network architecture

1) Transformer encoder: For converting an audio sequence X into a sequence of hidden representations, we build a Transformer-based encoder model [60] consisting of a stack of N_{enc} identical blocks:

$$H = \mathrm{TransformerEncoder}(X), \tag{1}$$

where $H = (\mathbf{h}_t \in \mathbb{R}^{d_{\mathsf{model}}} | t = 1, \dots, T)$, and $\mathbf{h}_t$ is a d_{model}-dimensional hidden representation at index t.

Given an audio sequence X, the encoder first applies 2D convolution $\mathrm{Conv2D}(\cdot)$ to down-sample the input sequence length from T' to T $(= T'/4)$ [61]. Positional encodings $\mathrm{PosEnc}(\cdot)$ are then added to each frame of the down-sampled sequence, which results in an initial hidden sequence:

$$H^{(0)} = \mathrm{Conv2D}(X) + \mathrm{PosEnc}(T). \tag{2}$$

The i-th encoder block outputs a sequence of hidden representations $H^{(i)} = (\mathbf{h}_t^{(i)} \in \mathbb{R}^{d_{\mathsf{model}}} | t = 1, \dots, T)$ as

$$\bar{H}^{(i)} = \mathrm{LayerNorm}(H^{(i-1)}), \tag{3}$$
$$\tilde{H}^{(i)} = H^{(i-1)} + \mathrm{SelfAttention}(\bar{H}^{(i)}), \tag{4}$$
$$H^{(i)} = \tilde{H}^{(i)} + \mathrm{FeedForward}(\mathrm{LayerNorm}(\tilde{H}^{(i)})), \tag{5}$$

where $i \in \{1, \dots, N_{\mathsf{enc}}\}$, and $\mathrm{LayerNorm}(\cdot)$, $\mathrm{SelfAttention}(\cdot)$, and $\mathrm{FeedForward}(\cdot)$ indicate layer normalization, multi-head self-attention, and feed-forward network, respectively. The final sequence H is obtained by normalizing the output of the last encoder block, i.e., $H = \mathrm{LayerNorm}(H^{(N_{\mathsf{enc}})})$.

2) Conformer encoder: Besides the Transformer encoder, we construct a model based on the Conformer-based encoder [12] consisting of a stack of N_{enc} identical blocks:

$$H = \mathrm{ConformerEncoder}(X). \tag{6}$$

Conformer is a variant of Transformer-based encoder architecture augmented with convolution to increase the capability for capturing local feature patterns [12], which has been shown to be more effective than standard Transformers on various speech processing tasks [62].

The computation in each Conformer encoder block can be defined by modifying the encoder steps in Transformer, where Eqs. (4) and (5) are replaced with

$$\bar{\tilde{H}}^{(i)} = H^{(i-1)} + \frac{1}{2}\mathrm{FeedForward}(\bar{H}^{(i)}), \tag{7}$$
$$\tilde{H}^{(i)} = \bar{\tilde{H}}^{(i)} + \mathrm{SelfAttention}(\mathrm{LayerNorm}(\bar{\tilde{H}}^{(i)})), \tag{8}$$
$$\tilde{\tilde{H}}^{(i)} = \tilde{H}^{(i)} + \mathrm{Conv}(\mathrm{LayerNorm}(\tilde{H}^{(i)})), \tag{9}$$
$$H^{(i)} = \tilde{\tilde{H}}^{(i)} + \frac{1}{2}\mathrm{FeedForward}(\mathrm{LayerNorm}(\tilde{\tilde{H}}^{(i)})). \tag{10}$$

In addition to the self-attention layer, Conformer introduces a module $\mathrm{Conv}(\cdot)$ based on depthwise separable convolution [63]. The convolution module consists of point-wise convolution, gated linear unit activation, 1D depth-wise convolution, batch normalization, Swish activation, and point-wise convolution. Unlike Transformer, each Conformer block adopts relative positional encoding [64] for the self-attention layer, which enables the model to increase the robustness against different input lengths. Moreover, Conformer employs the Macaron Net-style structure [65], where the original feed-forward layer (Eq. (5)) is replaced with two half-step feed-forward layers (Eqs. (7) and (10)).

B. Connectionist temporal classification (CTC)

CTC [3], [6] optimizes end-to-end ASR by training a model to find monotonic alignments between an input sequence X and target sequence Y. To align the sequences at the frame level, Y is augmented by adding an additional blank token ϵ and allowing repetitions of the same token, which results in a CTC alignment $Z = (z_t \in \mathcal{V} \cup \{\epsilon\} | t = 1, \dots, T)$. Assuming the conditional independence of frame-wise token predictions, CTC models the probability $P(Z|X)$ as the product of token emission probabilities:

$$P(Z|X) = \prod_{t=1}^{T} P(z_t | z_1, \dots, z_{t-1}, X), \tag{11}$$
$$\approx \prod_{t=1}^{T} P(z_t | X), \tag{12}$$

where $P(z_t|X)$ is a probability density function of the tokens and is obtained by applying a linear projection layer and a softmax layer to the encoded sequence H from Eq. (1) or (6).

For a given target sequence, there exist several possible alignments, depending on the position of the blank tokens and the number of repeated tokens. Let $\mathcal{B}$ be a collapsing function that maps a CTC alignment Z to a target sequence Y, which is performed by suppressing repeated tokens and removing blank tokens. With the collapsing function, CTC calculates the probability $P(Y|X)$ by marginalizing Eq. (12) over CTC alignments as

$$P(Y|X) = \sum_{Z \in \mathcal{B}^{-1}(Y)} P(Z|X), \tag{13}$$

where the inverse function $\mathcal{B}^{-1}(Y)$ returns a set of CTC alignments that are compatible with Y. While Eq. (13) has to deal with all possible Z, it is efficiently computed via dynamic programming (e.g., forward-backward algorithm) [6].

Given a pair of input and output sequences (X, Y), a model is trained to minimize the CTC loss defined by the negative log-likelihood of Eq. (13):

$$\mathcal{L} = -\log P(Y|X). \tag{14}$$

C. Semi-supervised ASR with pseudo-labeling

The goal of semi-supervised ASR, in this work, is to exploit a large amount of unlabeled (audio-only) data to enhance a pre-trained ASR model in a self-training manner. To this end, we focus on an approach based on pseudo-labeling [33], [35], which is described in two steps: 1) the supervised training of a seed end-to-end ASR model, and 2) the semi-supervised training of the seed model using unlabeled data.

1) Supervised training of a seed model: A seed model P_θ with parameters θ is first trained on labeled data $\mathcal{D}_{\mathsf{lab}} = \{(X_n, Y_n)|n = 1, \dots, N\}$, using the CTC loss from Eq. (14):

$$\mathcal{L}_{\mathsf{lab}}(\theta) = -\log P_\theta(Y_n|A(X_n)), \tag{15}$$

where $A(\cdot)$ denotes a data augmentation for improving generalization of the model, here SpecAugment [55].

2) Semi-supervised training with pseudo-labels: The seed model is then used to generate pseudo-labels for unlabeled data $\mathcal{D}_{\mathsf{unlab}} = \{X_{N+m}|m = 1, \dots, M\}$.[2] For each unlabeled sample, pseudo-labels $\hat{Y}_{N+m}$ are generated as

$$\hat{Y}_{N+m} = \underset{Y \in \mathcal{V}^*}{\mathrm{argmax}} \{\log P_\theta(Y|X_{N+m}) + \gamma \log P_{\mathsf{lm}}(Y)\}, \tag{16}$$

where argmax is performed using an external LM P_{lm} and beam-search decoding, and γ is the LM weight. With the pseudo-labels from Eq. (16), the loss for the unlabeled data is calculated in the same manner as Eq. (15):

$$\mathcal{L}_{\mathsf{unlab}}(\theta) = -\log P_\theta(\hat{Y}_{N+m}|A(X_{N+m})). \tag{17}$$

Finally, using both $\mathcal{D}_{\mathsf{lab}}$ and $\mathcal{D}_{\mathsf{unlab}}$, the seed model is further trained on the combined objective of $\mathcal{L}_{\mathsf{lab}}$ and $\mathcal{L}_{\mathsf{unlab}}$.

IV. Momentum Pseudo-Labeling

In this section, we explain our momentum pseudo-labeling (MPL) for semi-supervised ASR [47], [48]. The overall process of MPL is shown in Algorithm 1, which trains a pair of *online* and *offline* models that interact and learn from each other. Let us define the online and offline models as P_ξ and P_ϕ, with model parameters ξ and ϕ, respectively. Both models are initialized with the seed model P_θ trained as in Section III-C1.

[2]Filtering out pseudo-labels of low quality is an effective technique for semi-supervised ASR [41], [43]. It is particularly important for sequence-to-sequence models, where their autoregressive structure is prone to looping and early stopping issues during inference [35], [66] and likely to generate severely erroneous pseudo-labels. As CTC-based models are known to be robust against length issues [67], we do not perform any label filtering in this work.

Algorithm 1 Momentum pseudo-labeling

Input:
$\mathcal{D}_{\mathsf{lab}}, \mathcal{D}_{\mathsf{unlab}}$ ▷ labeled and unlabeled data
$\mathcal{A}$ ▷ an ASR model architecture
α ▷ a momentum coefficient

Train a seed model P_θ with architecture $\mathcal{A}$ on $\mathcal{D}_{\mathsf{lab}}$ using Eq. (15)
Initialize an online model P_ξ and an offline model P_ϕ with P_θ
for epoch $= 1, \dots, E_{\mathsf{mpl}}$ **do**
  for all $S \in \mathcal{D}_{\mathsf{lab}} \cup \mathcal{D}_{\mathsf{unlab}}$ **do**
    Obtain $X \sim S$
    Obtain $Y = \begin{cases} Y \sim S & (S \in \mathcal{D}_{\mathsf{lab}}) \\ \hat{Y} \sim P_\phi(Y|X) & (S \in \mathcal{D}_{\mathsf{unlab}}) \end{cases}$
    Compute loss $\mathcal{L}$ for $P_\xi(Y|X)$ as in Eq. (15)
    Update ξ using $\nabla_\xi \mathcal{L}$
    Update $\phi \leftarrow \alpha\phi + (1-\alpha)\xi$
  end for
end for
return P_ξ ▷ online model is returned for final evaluation

A. Online model training

On unlabeled sample $X_{N+m} \in \mathcal{D}_{\mathsf{unlab}}$, the online model is trained using pseudo-labels $\hat{Y}_{N+m}$ generated on the fly by the offline model:

$$\hat{Y}_{N+m} = \underset{Y \in \mathcal{V}^*}{\mathrm{argmax}} P_\phi(Y|X_{N+m}), \tag{18}$$

where argmax is performed based on the best path decoding of CTC [6]. Specifically, the most probable tokens $\hat{Z}$ are selected at each frame, and an output sequence is obtained using the collapsing function, i.e., $\hat{Y} = \mathcal{B}(\hat{Z})$. Note that Eq. (18) differs from Eq. (16) in that we use neither LM nor beam-search decoding.

With unlabeled data $X_{N+m} \in \mathcal{D}_{\mathsf{unlab}}$ and the corresponding pseudo-labels from Eq. (18), the semi-supervised loss of the online model is defined in the same manner as Eq. (17):

$$\mathcal{L}_{\mathsf{unlab}}(\xi) = -\log P_\xi(\hat{Y}_{N+m}|A(X_{N+m})), \tag{19}$$

which is maximized via a gradient descent optimization. In Eq. (19), we apply the data augmentation to an unlabeled input, aiming to provide the online model with training signals to learn robustly from the noisy input [36], [40]. In Section VI-D, we show that data augmentation is an important factor of MPL.

Assuming labeled data $\mathcal{D}_{\mathsf{lab}}$ is available during the semi-supervised process, we also use the supervised loss $\mathcal{L}_{\mathsf{lab}}(\xi)$ calculated similarly as in Eq. (15), which helps to stabilize the online model training as it learns from unlabeled data. Using $\mathcal{D}_{\mathsf{lab}}$ and $\mathcal{D}_{\mathsf{unlab}}$, the online model is trained using the combined objective of $\mathcal{L}_{\mathsf{lab}}(\xi)$ and $\mathcal{L}_{\mathsf{unlab}}(\xi)$. Note that in Section VI-B, we demonstrate that MPL is yet stable and effective even when trained solely on unlabeled data, i.e., trained only with $\mathcal{L}_{\mathsf{unlab}}(\xi)$.

B. Offline model training

After every update of the online model, the offline model accumulates parameters of the online model via

$$\phi \leftarrow \alpha\phi + (1-\alpha)\xi, \tag{20}$$

an exponential moving average with a momentum coefficient $\alpha \in [0, 1]$. This momentum update makes the offline model evolve more smoothly than the online model. We can thus

control the change in pseudo-labels generated on the fly by the offline model at each training step. This is important to prevent pseudo-labels from deviating too quickly from the initial labels generated by the seed model and to avoid collapsing to a trivial solution. Indeed, we empirically observe that training is prone to collapse (emitting only blank tokens for unlabeled data) for $\alpha = 0.0$, in which case the online and offline models share parameters and the online model is trained with self-generated pseudo-labels as in [40]. The problem is prominent when there is a domain mismatch between labeled and unlabeled data, as is often the case in real-world deployment. At the other end of the spectrum, when $\alpha = 1.0$, this approach becomes similar to standard pseudo-labeling [35] as described in Section III-C, where the offline model is never updated, and the online model is trained on fixed pseudo-labels generated from the seed model. This can stabilize the semi-supervised training, at the cost of leaving no room for improving pseudo-labels and limiting the improvement of the online model. We demonstrate the effectiveness of the momentum update in Section VI-B.

After training with MPL, both the online and offline models can be used for evaluation, although we use the online model as our default. We compare and analyze the performance of these models in Section VI-C.

C. Tuning the momentum coefficient

Instead of directly tuning α in Eq. (20), we design a more intuitive method for deriving an appropriate value of α. Based on Eq. (20), the parameters of the offline model after K iterations can be written as

$$\phi^{(K)} = \alpha^K \phi^{(0)} + (1-\alpha) \sum_{k=1}^{K} \alpha^{K-k} \xi^{(k)}, \tag{21}$$

where $\phi^{(k)}$ and $\xi^{(k)}$ denote the parameters of each model at the k-th iteration, and $\phi^{(0)} = \xi^{(0)} = \theta$. We here assume that it is important to retain some influence of the seed model to stabilize the pseudo-label generation. As a measure of this influence, we focus on the term $\alpha^K \phi^{(0)}$ in Eq. (21) and define a weight w of the seed model in $\phi^{(K)}$ as

$$w = \alpha^K, \tag{22}$$

where we consider K as the number of iterations (i.e., mini-batches) in a training epoch. As K can often be in the thousands, small changes in α lead to huge differences in w (e.g., $0.999^{3000} \ll 0.9997^{3000}$), requiring small adjustments on α for different amounts of training data. Instead of directly tuning α for the momentum update, we propose to tune the weight w, which can be regarded as the proportion of the seed model parameters retained after a training epoch. Given w and K, α is calculated as

$$\alpha = e^{(1/K)\log w}. \tag{23}$$

By controlling the update through w, we expect MPL to be less affected by the amount of training data, which we examine in Section VI-B.

D. Adopting Conformer for MPL training

To further enhance the MPL performance, we investigate utilizing the Conformer-based architecture [12], which is expected to improve overall ASR performance and thus enable a model to generate accurate pseudo-labels. While Conformer-based models have achieved outstanding ASR performance compared with standard Transformers [62], we empirically observe that Conformer suffers from poor generalization from labeled data to unlabeled data. A similar issue has been reported in other ASR tasks [68]–[70]. Simply adopting Conformer for MPL makes the training become unstable and diverge easily, especially when a domain mismatch exists between labeled and unlabeled data.

We assume that such a problem comes from unreliable statistics computed and used by batch normalization (BN) [71] in the convolution module (in Eq. (9)). As the seed model is first trained on labeled data only, the mean and variance estimated in BN are fitted to the statistics of $\mathcal{D}_{\mathsf{lab}}$. When the model is then further trained on the combined $\mathcal{D}_{\mathsf{lab}}$ and $\mathcal{D}_{\mathsf{unlab}}$ via MPL, the data variation becomes large among mini-batches, which leads to making BN unstable [72]. Hence, we consider replacing BN with group normalization (GN) [73] in the convolution module, as it has been shown effective for Conformer-based streaming ASR [68]. GN divides feature maps into groups and normalizes the features within each group, which makes the training less dependent on the variations across mini-batches. This is found critical for stabilizing the Conformer-based MPL training, as carefully investigated in Section VI-E1.

E. Exploiting LM knowledge for MPL training

While using an external LM and beam-search decoding has been shown to be effective for generating pseudo-labels with high-quality [35], [38], [40], it is too computationally intensive to be adopted for MPL due to the on-the-fly label generation. To mitigate this limitation, we consider performing the standard pseudo-labeling (PL) training (as described in Section III-C2) prior to MPL. With this combination of PL and MPL, LM knowledge is implicitly transferred to the seed model, providing the MPL training with a better initialization for generating higher-quality pseudo-labels. Moreover, by avoiding the explicit LM usage during the MPL training, we can prevent the ASR model from over-fitting to the LM training text data, which often degrades the generalization capability of the model [42], [43]. In addition to PL, we investigate iterative pseudo-labeling (IPL) [39], which extends PL by continuously training a model with periodic regeneration of pseudo-labels.

Algorithm 2 shows the proposed MPL training with the initialization strategy based on PL or IPL. In the beginning, a seed model is trained using a labeled set as in Section III-C1 (line 1). Then, the seed model is further trained via PL or IPL with LM and beam-search decoding (lines 3–11). Here, we denote I_{ipl} as the number of iterations (pseudo-label updates), and E_{ipl} as the number of epochs trained in each iteration. Note that this process becomes PL [35] when $I_{\mathsf{ipl}} = 1$ and IPL [39] when $I_{\mathsf{ipl}} > 1$. Finally, the enhanced seed model is

Algorithm 2 Incorporating LM knowledge into MPL

Input:
$\mathcal{D}_{\mathsf{lab}}, \mathcal{D}_{\mathsf{unlab}}$ ▷ labeled and unlabeled data
$\mathcal{A}$ ▷ an ASR model architecture
α ▷ a momentum coefficient

1: Train a seed model P_θ with architecture $\mathcal{A}$ on $\mathcal{D}_{\mathsf{lab}}$ using Eq. (15)
2: `# Iterative pseudo-labeling`
3: **for** $i = 1, \ldots, I_{\mathsf{ipl}}$ **do**
4: Generate $\hat{\mathcal{D}}_{\mathsf{unlab}} = \{(X_{N+m}, \hat{Y}_{N+m}) | X_{N+m} \in \mathcal{D}_{\mathsf{unlab}}\}$, using Eq. (16)
5: **for** epoch $= 1, \ldots, E_{\mathsf{ipl}}$ **do**
6: **for all** $(X, Y) \in \mathcal{D}_{\mathsf{lab}} \cup \hat{\mathcal{D}}_{\mathsf{unlab}}$ **do**
7: Compute loss $\mathcal{L}$ for $P_\theta(Y|X)$ with Eq. (15)
8: Update θ using $\nabla_\theta \mathcal{L}$
9: **end for**
10: **end for**
11: **end for**
12: `# Momentum pseudo-labeling`
13: Initialize an online model P_ξ and an offline model P_ϕ with P_θ
14: **for** epoch $= 1, \ldots, E_{\mathsf{mpl}}$ **do**
15: **for all** $S \in \mathcal{D}_{\mathsf{lab}} \cup \mathcal{D}_{\mathsf{unlab}}$ **do**
16: Obtain $X \sim S$
17: Obtain $Y = \begin{cases} Y \sim S & (S \in \mathcal{D}_{\mathsf{lab}}) \\ \hat{Y} \sim P_\phi(Y|X) & (S \in \mathcal{D}_{\mathsf{unlab}}) \end{cases}$
18: Compute loss $\mathcal{L}$ for $P_\xi(Y|X)$ as in Eq. (15)
19: Update ξ using $\nabla_\xi \mathcal{L}$
20: Update $\phi \leftarrow \alpha\phi + (1-\alpha)\xi$
21: **end for**
22: **end for**
23: **return** P_ξ ▷ online model is returned for final evaluation

used to initialize the online and offline models for MPL (lines 13–22). The MPL training lasts E_{mpl} epochs.

V. Experimental setting

A. Data

The experiments were carried out using the LibriSpeech (LS) [74] and TEDLIUM3 (TED3) [75] datasets. LS is a corpus of read English speech, containing 960 hours of training data (split into *train-clean-100*, *train-clean-360*, and *train-other-500*). We also used the 10-hour Libri-Light (LL) training data (*train-10h*) [16], which is a low-resource subset extracted from the LS training data. TED3 is a corpus of English Ted Talks consisting of 450 hours of training data (*train-ted3*). For each dataset, we used the standard validation and test sets for tuning hyper-parameters and evaluating performance, respectively. As input speech features, we extracted 80 mel-scale filterbank coefficients with three-dimensional pitch features using Kaldi [76]. For text tokenization, we used SentencePiece [77] to construct a 1k subword vocabulary, which was extracted from either *train-clean-100* or *train-10h* transcriptions, depending on the semi-supervised setting.

B. Semi-supervised settings

We simulated several semi-supervised settings, where either the *train-clean-100* (LS-100), *train-10h* (LL-10), or *train-clean-460* (LS-460) set is regarded as labeled. With a seed model trained on LS-100, we considered three settings with different unlabeled sets:

- **LS-100/LS-360**, an in-domain setting with *train-clean-360* (LS-360);
- **LS-100/LS-860**, an in-domain setting with *train-clean-360* and *train-other-500* (LS-860); and
- **LS-100/TED3**, an out-of-domain setting with *train-ted3*.

In addition, with a seed model trained on LL-10, we considered three low-resource settings with different unlabeled sets:

- **LL-10/LS-360**, an in-domain setting with *train-clean-360* (LS-360);
- **LL-10/LS-960**[3], an in-domain setting with *train-clean-100*, *train-clean-360*, and *train-other-500* (LS-960); and
- **LL-10/TED3**, an out-of-domain setting with *train-ted3*.

We also considered two settings with more labeled data, using LS-100 and LS-360 (LS-460) for training a seed model:

- **LS-460/LS-500**, an in-domain setting with unlabeled *train-other-500* (LS-500); and
- **LS-460/TED3**, an out-of-domain setting with unlabeled *train-ted3*.

TABLE I
Number of mini-batches K and momentum coefficient α in each setting. Given $w = 0.5$ and K, α is calculated from Eq. (23)

Setting	K	α
LS-100/LS-360	1528	0.99955
LS-100/LS-860	3175	0.99978
LS-100/TED3	2077	0.99967
LL-10/LS-360	1230	0.99943
LL-10/LS-960	3207	0.99978
LL-10/TED3	1779	0.99961
LS-460/LS-500	3175	0.99978
LS-460/TED3	3274	0.99979

C. Model architecture

As an ASR model, we trained the Transformer [60] or Conformer [12]-based encoder architecture described in Section III-A1 or III-A2, implemented in ESPnet [78]. The model consisted of the Conv2D layer (in Eq. (2)) followed by a stack of 12 encoder blocks ($N_{\mathsf{enc}} = 12$). The Conv2D layer down-samples the input length by a factor of 4, using two 2D convolution layers with 256 channels, a kernel size of 3×3, and a stride size of 2. In the multi-head self-attention module (in Eqs. (4) and (8)), the number of heads d_{h} and dimension of a self-attention layer d_{model} were set to 4 and 256, respectively. The dimension of the feed-forward network d_{ff} (in Eqs. (5), (7), and (10)) was set to 2048. For the convolution module of Conformer (in Eq. (9)), we used a kernel size of 31. The number of groups was set to 8 for group normalization when it was used as a replacement for batch normalization in the convolution module.

D. Training configuration

The seed model was trained for 150 epochs using the Adam optimizer [79] with $\beta_1 = 0.9$, $\beta_2 = 0.98$, and $\epsilon = 10^{-9}$. We used Noam learning rate scheduling [60] with 25k warmup

[3]Note that LS-960 contains the LL-10 audios, so the LL-10 utterances are included both as labeled and unlabeled samples, and only the LS-960 data other than LL-10 should be considered as truly unlabeled.

steps and a learning rate factor of 5.0. The semi-supervised training lasted up to 200 epochs, where the gradient-based optimization was done by using the Adam optimizer with an initial learning rate of 10^{-3}, $\beta_1 = 0.9$, $\beta_2 = 0.999$, and $\epsilon = 10^{-8}$. IPL was performed by iterating PL for a maximum of 8 times ($I_{\text{ipl}} \leq 8$), where the model was trained for 25 epochs ($E_{\text{ipl}} = 25$) in each iteration. After each iteration of IPL, we averaged model parameters over the last 5 checkpoints to stabilize the pseudo-label generation. For the momentum update of MPL (in Eq. (20)), we used $w = 0.5$ for all the semi-supervised settings, and Table I lists an actual value of the momentum coefficient α used in each setting. For the training of all models, we used SpecAugment [55] as data augmentation.

E. Decoding configuration

For evaluation, a final model was obtained by averaging model parameters over 10 checkpoints with the best validation performance [60]. We trained an LM consisting of 4 long short-term memory (LSTM) layers with 2048 units, using the LS-100 or LL-10 transcriptions combined with the external text data provided by LibriSpeech [74]. For decoding with the LM, we adopted a frame-synchronous CTC prefix beam search algorithm [80], [81], where we used a beam-size of 20, a score-based pruning threshold of 14.0, an LM weight of 1.0, and an insertion bonus factor of 2.0. For decoding without the LM, we performed the best path decoding of CTC [6].

F. Evaluation metrics

We used the word error rate (WER) to measure the ASR performance. For evaluating the performance of semi-supervised training, we measured the WER recovery rate (WRR) [35], [82]. WRR compares WERs of the oracle model (trained using ground-truth transcriptions for the unlabeled data as well) and the semi-supervised model by calculating the ratio between their absolute reductions from the seed model's WER:

$$\text{WRR}[\%] = \frac{\text{WER}_{\text{seed}} - \text{WER}_{\text{semi-supervised}}}{\text{WER}_{\text{seed}} - \text{WER}_{\text{oracle}}}, \qquad (24)$$

where WER_* denotes WER for each model.

VI. Results and Discussion

In this section, we report and discuss results obtained from our semi-supervised ASR experiments. First, to verify the effectiveness of the proposed MPL, we perform some basic analyses based on the Transformer-based models. Then, we show results for further improving MPL, using the Conformer architecture and additional training/decoding techniques.

A. MPL results using Transformer

1) In-domain settings: Table II shows results on the in-domain LS settings, comparing PL [35] (from Section III-C) and the proposed MPL. Note that the MPL results also appear in our previous paper [48]. The oracle results were obtained by fine-tuning the seed model using ground-truth labels for both the labeled source and target training sets. Looking at results in the LS-360 setting (`A*`), both PL and MPL led to a significant improvement over the seed model (`A1`,`A2` vs. `S1`). When decoded without the external LM, PL resulted in better performance than MPL (`A1` vs. `A2`), benefiting from high-quality pseudo-labels generated using the LM. However, when decoded with the LM, the performance gain was larger for MPL, achieving much lower WERs on the *other* sets with similar WRRs to those obtained by decoding without the LM. PL, in contrast, had smaller improvement with degraded WRRs, which indicates PL is likely to fit to LM knowledge, as reported in [42], [43], and have less variations in the hypotheses during the beam-search decoding.

In the LS-860 setting (`B*`) with more unlabeled data, MPL again outperformed the seed model with the same range of WRR as was observed in the LS-360 setting (`A2` vs. `B2`), demonstrating its scalability with respect to the amount of unlabeled data. While PL was also effective in this setting, the gain from the seed model was smaller than that obtained from the LS-360 setting (`A1` vs. `B1`); on the *other* sets, especially, the WRRs of PL dropped by an absolute difference of over 10%. MPL was capable of keeping high WRRs on the *other* sets, successfully increasing the generalization ability of the model. MPL greatly benefited from decoding with the external LM and achieved better results than those obtained with PL.

2) Out-of-domain setting: Table III lists results on the out-of-domain TED3 setting. Note that the MPL results also appear in our previous paper [48]. PL resulted in a modest improvement over the seed model, while the gain was more substantial for MPL (`C1` vs. `C2`). As there is a domain mismatch between the LM training text and the actual TED3 transcriptions, PL was less effective at learning from the out-of-domain unlabeled data. Moreover, the LM decoding led to lowering the WRR of PL, indicating that the model was prone to over-fitting to LM knowledge. MPL, on the other hand, took great advantage of decoding with the LM while achieving as high a WRR as the result decoded without the LM.

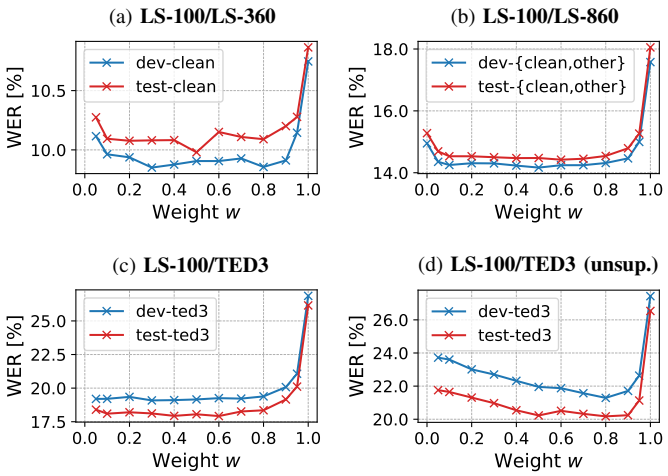

Fig. 1. Influence of momentum update weight w on WER.

B. Effectiveness of w for tuning the momentum update

Figure 1 shows the performance of MPL depending on the weight w (defined in Eq. (22)) used to derive α in the

TABLE II
WERs [%] AND WRRs [%] ON IN-DOMAIN LIBRISPEECH (LS) SETTINGS. ALL MODELS WERE TRAINED USING TRANSFORMER (TRF) ENCODER. THE RESULTS ARE DIVIDED INTO TWO SECTIONS: WHETHER THE LM WITH BEAM-SEARCH DECODING WAS APPLIED IN THE FINAL EVALUATION OR NOT.

			Decoding without LM						Decoding with LM					
			Dev WER		**Test WER**		**Test WRR**		**Dev WER**		**Test WER**		**Test WRR**	
Setting	**Method**	**Init.**	clean	other	clean	other	clean	other	clean	other	clean	other	clean	other
LS-100	S1 seed (Trf)	–	12.2	30.0	12.9	31.1	0.0	0.0	5.8	17.8	6.3	18.9	0.0	0.0
LS-100 / LS-360	A1 PL	S1	**7.6**	**20.8**	**8.0**	**21.1**	**77.1**	**82.9**	**4.6**	13.7	**5.0**	13.9	**48.4**	64.3
	A2 MPL	S1	8.7	21.4	9.0	21.7	61.5	77.9	4.8	**13.0**	5.1	**13.1**	42.5	**74.9**
	A3 oracle	S1	6.2	19.2	6.5	19.1	100.0	100.0	3.3	10.9	3.6	11.1	100.0	100.0
LS-100 / LS-860	B1 PL	S1	**7.1**	17.9	**7.2**	18.4	**74.0**	71.4	**4.4**	12.6	**4.8**	13.3	46.1	50.3
	B2 MPL	S1	8.1	**16.5**	8.3	**16.8**	59.9	**80.2**	4.6	**9.7**	**4.8**	**10.1**	**46.5**	**78.8**
	B3 oracle	S1	5.1	13.1	5.2	13.3	100.0	100.0	2.8	7.5	3.0	7.8	100.0	100.0

TABLE III
WERs [%] AND WRRs [%] ON OUT-OF-DOMAIN TEDLIUM3 (TED3) SETTING. ALL MODELS WERE TRAINED USING TRANSFORMER (TRF) ENCODER.

			Decoding without LM			Decoding with LM		
Setting	**Method**	**Init.**	**Dev WER**	**Test WER**	**Test WRR**	**Dev WER**	**Test WER**	**Test WRR**
LS-100	S1 seed (Trf)	–	31.2	32.0	0.0	23.4	23.2	0.0
LS-100 / TED3	C1 PL	S1	21.1	20.6	59.2	18.6	18.6	33.6
	C2 MPL	S1	**18.4**	**17.0**	**78.0**	**14.9**	**13.3**	**73.1**
	C3 oracle	S1	12.7	12.8	100.0	10.1	9.6	100.0

momentum update (in Eq. (20)). Note that the figures are reproduced from our previous paper [47]. We observed a similar trend among the curves in different semi-supervised settings (Figs. 1(a), 1(b), and 1(c)). WERs increased as w was set closer to 0.0. When $w = 0.0$, which is a similar approach to [40], the training was likely to be unstable and failed under the LS-360 and TED3 conditions, as illustrated by the learning curves shown in Fig. 2. This indicates the importance of retaining the influence of the seed model to stabilize learning from unlabeled data. However, depending too much on the seed model (i.e., setting w closer to 1.0) also worsened WERs. Larger w slows down the progress of the offline model, causing MPL to become more like standard PL [35] but without an LM.

Figure 1(d) shows results under an extreme condition, where not only a domain mismatch exists between the seed model and unlabeled data, but labeled data is not used during the semi-supervised process (i.e., training the online model with $\mathcal{L}_{\text{unlab}}(\xi)$ only). Compared to the other settings, the performance was more sensitive to the change in w, but the overall trend was similar.

In general, the proposed tuning method with w effectively controlled the momentum update in all settings. It provides a more intuitive guide for tuning α, taking the amount of data into account. Based on the validation results mainly on the LS-860 and TED3 settings, we set $w = 0.5$ for all the semi-supervised settings.

C. Online model vs. offline model

Table IV compares results obtained from the online and offline models after the MPL training in LS-100/LS-360. Here,

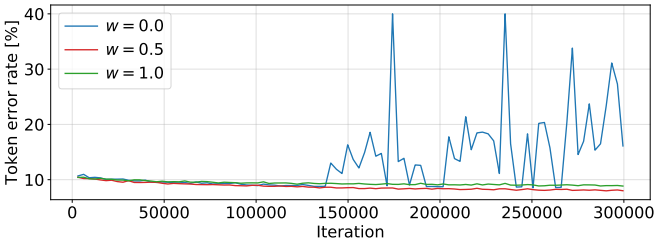

Fig. 2. Learning curves of MPL with different weights w in LS-100/LS-360.

TABLE IV
WERs [%] OF ONLINE AND OFFLINE MODELS IN LS-100/LS-360 SETTING. DECODED WITHOUT LM.

Avg. metric	**Model**	**Dev WER**	**Test WER**
last1	online	9.2 / 22.6	9.4 / 22.8
	offline	**8.7** / 21.7	9.1 / 22.1
val10	online	**8.7** / **21.4**	**9.0** / **21.7**
	offline	**8.7** / 21.6	**9.0** / **21.7**

last1 indicates evaluating each model from the last checkpoint and *val10* from averaging model parameters over 10 checkpoints with the best validation performance. Without the checkpoint averaging technique, the offline model gave better performance than the online model. Being an exponential moving average of the online model parameters over the MPL training (cf. Eq. (20)), the offline model naturally benefited from the model ensembling, as it has been shown effective in [83]. However, with checkpoint averaging, the performance of both models improved and the gap was reduced to almost

TABLE V
WERs [%] AND WRRs [%] FOR MPL WITHOUT DATA AUGMENTATION. † INDICATES WITHOUT SPECAUGMENT. DECODED WITHOUT LM.

Setting	Method	Dev WER	Test WER	Test WRR
LS-100 / LS-360	MPL (A2)	8.7 / 21.4	9.0 / 21.7	61.5 / 77.9
	MPL†	10.6 / 27.2	11.0 / 28.2	37.9 / 43.3
	oracle†	7.5 / 23.8	7.8 / 24.3	100.0 / 100.0
LS-100 / LS-860	MPL (B2)	8.1 / 16.5	8.3 / 16.8	59.9 / 80.2
	MPL†	10.2 / 23.2	10.3 / 23.8	36.4 / 48.7
	oracle†	5.8 / 15.8	5.8 / 16.1	100.0 / 100.0
LS-100 / TED3	MPL (C2)	18.4	17.0	78.0
	MPL†	22.8	22.2	55.5
	oracle†	13.0	12.6	100.0

TABLE VI
VALIDATION WERs [%] OF SEED MODELS TRAINED ON LABELED LS-100. FOR THE CONFORMER-BASED MODELS, WE EXPLORED DIFFERENT NORMALIZATION METHODS FOR THE CONVOLUTION MODULE.

Model	Norm. type	LibriSpeech dev-clean	LibriSpeech dev-other	TED3 Dev
Transformer (S1)	–	12.2	30.0	31.2
Conformer	Batch	8.6	23.1	27.3
	Instance	8.9	23.5	27.1
	Group	**8.4**	**22.5**	**26.4**
	Layer	**8.4**	22.9	26.9

none. As the online model was slightly better on the development sets, we used it as our default for evaluation, which is contrary to previous work [46].

D. Importance of data augmentation

In Table V, we investigate the importance of applying SpecAugment during the MPL training (cf. Eq. (19)). Even without the augmentation, MPL led to a decent improvement over the seed model (S1 in Table II). However, WRRs significantly dropped compared to the results with SpecAugment (A2,B2,C2). Note that, for the models trained without the augmentation, we computed the WRR against an oracle model without the augmentation. Without SpecAugment, we observed that the training converged earlier, and MPL was less effective.

E. MPL results using Conformer

1) Investigation on normalization method: In Table VI, we compare WERs of seed models trained using the Transformer or the Conformer architecture. Note that similar results also appear in our previous paper [48]. For Conformer-based models, we investigated different normalization methods for the convolution module (in Eq. (9)), including {batch [71], instance [84], group [73], layer [85]} normalization ({BN, IN, GN, LN}). Note that IN and LN are the same as GN with group sizes 1 and 256 ($= d_{\text{model}}$), respectively. Comparing the two architectures, the Conformer-based models significantly improved over the Transformer-based model (S1 in Table II). Within the Conformer-based models, GN resulted in the best performance on both LS and TED3, and the 100-hour training data seemed to be too small to take advantage of BN. As normalizing across feature maps (i.e., IN, GN, and LN) achieved better performance than BN on the out-of-domain TED3 set, this indicates that BN led to lower generalization capability with unreliable statistics. Note that in [11], BN achieved better performance than the other normalization methods when another ASR model based on depth-wise separable convolution was trained on the labeled full 960-hour set of LS.

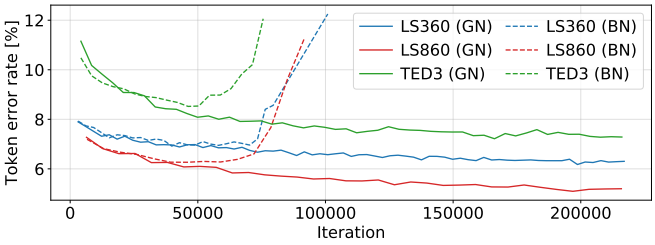

Fig. 3. Validation token error rate [%] of MPL training using Conformer with batch (dotted line) or group (solid line) normalization.

Figure 3 shows learning curves from MPL training using Conformer with BN or GN. The figure is reproduced from our previous paper [48]. In all semi-supervised settings, BN caused the training to become unstable. This was especially the case in the out-of-domain setting with TED3, where the model diverged more quickly than in the other settings. In contrast, GN successfully stabilized the MPL training with Conformer.

2) In-domain setting: Table VII lists results on the in-domain LS settings, comparing PL [35], IPL [39], and the proposed MPL. Note that similar results also appear in our previous paper [48]. Looking at the MPL results (X3,Y3), MPL led to a substantial improvement over the seed model (S2), effectively learning from unlabeled data using Conformer with GN. These Conformer results significantly outperformed those of Transformer-based MPL (A2,B2 from Table II). With pseudo-labels generated using the LM, PL [35] and IPL [39] achieved lower WERs on the *clean* sets than those obtained from MPL, and IPL resulted in better performance than MPL on the *other* sets as well (⋆1,⋆2 vs. ⋆3). However, when decoded with the LM, the performance gain was larger for MPL with only a slight decrease in WRRs compared with the decoding without LM, and MPL achieved much lower WERs on the *other* sets. PL and IPL, in contrast, had smaller improvement with degraded WRRs, as was observed in the Transformer results (Table II).

3) Out-of-domain setting: Table VIII shows MPL results on the TED3 setting. Note that similar results also appear in our previous paper [48]. Conformer with GN significantly improved MPL over the seed model and Transformer-based MPL (Z3 vs. S2,C2), successfully stabilizing training on the out-of-domain unlabeled data. PL and IPL led to decent improvements over the seed model, but the gain was more substantial for MPL (C1 vs. C2), which is consistent with the Transformer results (Table III). Moreover, PL and IPL had little gain from decoding with the LM, indicating the model was too fitted to the LM knowledge.

TABLE VII
WERs [%] AND WRRs [%] ON IN-DOMAIN LIBRISPEECH (LS) SETTINGS. ALL MODELS WERE TRAINED USING CONFORMER (CFM) ENCODER. ‡ INDICATES TRAINED FOR 100 EPOCHS.

			Decoding without LM						Decoding with LM					
			Dev WER		**Test WER**		**Test WRR**		**Dev WER**		**Test WER**		**Test WRR**	
Setting	**Method**	**Init.**	clean	other	clean	other	clean	other	clean	other	clean	other	clean	other
LS-100	S2 seed (Cfm)	–	8.4	22.5	8.6	23.3	0.0	0.0	5.2	15.2	5.5	16.0	0.0	0.0
LS-100 / LS-360	X1 PL	S2	5.7	15.9	6.1	15.8	64.6	76.0	4.3	11.4	4.5	11.8	40.6	62.2
	X2 IPL	S2	**5.4**	15.1	5.7	15.3	73.3	81.5	4.2	11.5	4.5	11.7	42.2	62.5
	X3 MPL	S2	6.1	16.0	6.6	15.8	52.3	76.4	4.5	11.2	4.7	**11.1**	34.7	71.7
	X4 MPL‡	X1@ep100	5.7	15.5	6.1	15.6	64.8	77.8	4.2	11.1	4.5	11.3	44.0	69.3
	X5 MPL‡	X2@ep100	5.5	**15.0**	**5.6**	**15.1**	**75.1**	**83.3**	**4.1**	**10.8**	**4.3**	**11.1**	**51.4**	**72.7**
	X6 oracle	S2	4.1	13.6	4.7	13.4	100.0	100.0	2.9	9.4	3.2	9.2	100.0	100.0
LS-100 / LS-860	Y1 PL	S2	5.4	13.9	5.7	14.2	57.8	62.3	4.0	10.5	4.2	10.7	43.0	53.7
	Y2 IPL	S2	**4.7**	11.5	**5.0**	11.7	**71.0**	79.3	4.1	9.7	4.4	10.2	36.2	58.9
	Y3 MPL	S2	5.7	12.2	6.2	12.2	48.1	76.4	4.1	8.5	4.4	8.7	36.5	74.6
	Y4 MPL‡	Y1@ep100	5.1	12.1	5.3	12.4	64.0	75.1	3.7	8.4	3.9	8.8	51.0	73.3
	Y5 MPL‡	Y2@ep100	**4.7**	**11.0**	**5.0**	**11.1**	70.0	**83.9**	**3.6**	**7.8**	**3.8**	**8.2**	**54.0**	**79.6**
	Y6 oracle	S2	3.3	9.0	3.5	8.7	100.0	100.0	2.4	6.1	2.5	6.2	100.0	100.0

TABLE VIII
WERs [%] AND WRRs [%] ON OUT-OF-DOMAIN TEDLIUM3 (TED3) SETTING. ALL MODELS WERE TRAINED USING CONFORMER (CFM) ENCODER.

			Decoding without LM			Decoding with LM		
Setting	**Method**	**Init.**	**Dev WER**	**Test WER**	**Test WRR**	**Dev WER**	**Test WER**	**Test WRR**
LS-100	S2 seed (Cfm)	–	26.4	26.5	0.0	21.3	21.1	0.0
LS-100 / TED3	Z1 PL	S2	18.9	18.5	51.3	16.9	17.0	33.5
	Z2 IPL	S2	16.8	16.8	62.2	16.6	16.9	34.2
	Z3 MPL	S2	15.1	13.9	81.0	12.7	**11.6**	**77.3**
	Z4 MPL‡	Z1@ep100	15.5	15.0	73.9	13.4	12.8	67.9
	Z5 MPL‡	Z2@ep100	**14.6**	**13.8**	**81.1**	**12.4**	12.0	73.8
	Z6 oracle	S2	10.4	10.9	100.0	8.6	8.8	100.0

F. Exploiting LM knowledge in MPL via PL or IPL

In Tables VII and VIII, ⋆4 and ⋆5 show results for applying MPL training after enhancing the seed model using PL and IPL, respectively (cf. Section IV-E). Note that we performed PL or IPL for 100 epochs and MPL for another 100 epochs to match the total training epochs of the other methods.

In the in-domain settings (Table VII), this initialization strategy provided MPL with distinct improvements (X3 vs. X4,X5 and Y3 vs. Y4,Y5). With the IPL-based initialization, MPL achieved the best overall performance on both of the settings with different amounts of unlabeled data (X5,Y5). When decoded with the LM, the improved MPL achieved higher WRRs than those obtained from PL and IPL (e.g., X2 vs. X5), preventing the model from over-fitting to LM knowledge but exploiting it to improve ASR performance.

In the out-of-domain setting (Table VIII), MPL further reduced WERs by using the initialization based on IPL (Z3 vs. Z5). However, the improvement was much smaller than those observed in the in-domain settings, and the standard MPL performed sufficiently well by decoding with the LM.

G. Adapting language model

To further improve the MPL result in the out-of-domain setting, we explore adapting the LM to the target domain (i.e., TED3). To this end, we made an attempt to make use of pseudo-labels, which were generated from *train-ted3* using the model obtained from MPL (Z3). The pseudo-labels were simply mixed with the LS training text for training an adapted LM. Table IX shows results decoded with different LMs, where a source LM is trained on the LS training text (the same LM used in the other experiments), and a target LM is trained on the external text-only data provided by TED3 [75]. With the adapted LM, MPL slightly improved over decoding with the source LM. The seed and oracle models also benefited from decoding with the adapted LM, where the gain was much larger than that of MPL. As the pseudo-labels are obtained as a result of MPL training, the adapted LM was less effective for MPL with the already-

TABLE IX
TEST WERs [%] AND WRRs [%] ON LS-100/TED3. DIFFERENT LMs WERE USED DURING DECODING.

	Source LM		Adapted LM		Target LM	
Method	**WER**	**WRR**	**WER**	**WRR**	**WER**	**WRR**
seed (S2)	21.1	0.0	20.8	0.0	19.1	0.0
MPL (Z3)	11.6	77.3	11.5	75.3	10.6	75.8
oracle (Z6)	8.8	100.0	8.4	100.0	7.9	100.0

TABLE X
WERs [%] AND WRRs [%] ON LOWER-RESOURCE IN-DOMAIN LS SETTINGS. ALL MODELS WERE TRAINED USING CONFORMER (CFM) ENCODER. ‡ INDICATES TRAINED FOR 100 EPOCHS.

			Decoding without LM						Decoding with LM					
			Dev WER		**Test WER**		**Test WRR**		**Dev WER**		**Test WER**		**Test WRR**	
Setting	**Method**	**Init.**	clean	other	clean	other	clean	other	clean	other	clean	other	clean	other
LL-10	S3 seed (Cfm)	–	34.3	50.6	34.8	51.3	0.0	0.0	22.4	38.1	22.2	39.2	0.0	0.0
LL-10 / LS-360	I1 PL	S3	21.4	35.4	21.7	36.1	44.9	42.4	15.2	28.1	15.4	29.3	36.6	34.3
	I2 MPL	S3	19.0	28.5	19.3	28.7	53.1	62.9	14.8	21.0	14.8	21.3	39.8	62.0
	I3 MPL‡	I1@ep100	**16.2**	**27.7**	**16.3**	**27.9**	**63.1**	**65.2**	**12.0**	**20.3**	**12.1**	**20.8**	**54.6**	**63.5**
	I4 oracle	S3	5.1	15.9	5.5	15.4	100.0	100.0	3.3	10.2	3.6	10.3	100.0	100.0
LL-10 / LS-960	J1 PL	S3	20.2	33.5	20.5	34.4	46.0	40.3	14.7	27.0	14.8	28.2	38.1	33.7
	J2 MPL	S3	17.9	23.6	18.1	24.3	53.7	64.5	14.5	17.6	14.4	18.1	40.0	64.7
	J3 MPL‡	J1@ep100	**13.0**	**20.1**	**13.2**	**20.3**	**69.3**	**74.1**	**10.2**	**14.7**	**10.5**	**15.1**	**60.1**	**74.0**
	J4 oracle	S3	3.5	9.3	3.7	9.5	100.0	100.0	2.4	6.1	2.7	6.6	100.0	100.0

TABLE XI
WERs [%] AND WRRs [%] ON LOWER-RESOURCE OUT-OF-DOMAIN TED3 SETTING. ALL MODELS WERE TRAINED USING CONFORMER (CFM) ENCODER.

			Decoding without LM			Decoding with LM		
Setting	**Method**	**Init.**	**Dev WER**	**Test WER**	**Test WRR**	**Dev WER**	**Test WER**	**Test WRR**
LL-10	S3 seed (Cfm)	–	54.9	56.6	0.0	45.4	47.5	0.0
LL-10 / TED3	K1 PL	S3	42.1	42.8	29.2	38.1	39.6	20.1
	K2 MPL	S3	31.0	28.6	59.5	26.2	24.1	59.6
	K3 MPL‡	K1@ep100	**29.4**	**27.6**	**61.7**	**25.1**	**23.3**	**61.5**
	K4 oracle	S3	9.4	9.5	100.0	8.2	8.1	100.0

acquired knowledge. Regarding LM training on pseudo-labels (uncertain ASR hypotheses), an effective approach has been proposed in [86], which trains a Transformer or LSTM-LM by calculating a Kullback–Leibler divergence loss against token-wise predictions of ASR confusion networks. We found this method challenging to apply to our framework, as this work focused on a CTC-based model with frame-wise predictions. Hence, future work should consider a better way for training an LM using pseudo-labels from a CTC-based ASR model.

H. MPL results in lower-resource settings

1) In-domain setting: Table X lists in-domain results where Conformer-based PL and MPL are applied to the LL-10 settings. With a fewer amount of labeled data, the seed model was inferior in quality, compared to the one trained on LS-100 (S2 vs. S3). In both of the settings, PL and MPL successfully improved the seed model (⋆1,⋆2 vs. S3). Even without using the LM, MPL achieved much lower WERs than PL (⋆1 vs. ⋆2), and PL resulted in a significant drop in WRRs compared to the previous experiments with more labeled data (I1 vs. X1 and J1 vs. Y1). While PL-based approaches often depend on the quality of a seed model, MPL managed to alleviate the problem by continuously improving the pseudo-label quality via the interaction between the online and offline models. Using PL as an initialization, MPL further improved the performance by exploiting the LM knowledge effectively.

2) Out-of-domain setting: Table XI shows results on the out-of-domain setting. While PL improved over the seed model, the gain was smaller when compared to the results from the in-domain settings (I1,J1 vs. K1). On the other hand, MPL performed better than PL and kept WRRs as high as the in-domain results (I2,J2 vs. K2). Even with the low-quality seed model, MPL enabled the model to train stably on the out-of-domain data, and the tuning of the momentum update (Section IV-C) worked robustly to the amount of labeled data. The PL-based initialization was also effective for improving MPL performance while maintaining the high WRRs when decoded with the LM (K3).

I. MPL results in higher-resource settings

1) In-domain setting: Table XII lists in-domain results where Conformer-based PL and MPL are evaluated on the LS-460 settings. With a larger amount of labeled data, the seed model had better quality than the models trained on less data (S2,S3 vs. S4). In both settings, PL and MPL improved over the seed model (L1,L2 vs. S4), and the higher-quality seed model led to better overall results compared to the LS-100 settings (Table VII). When decoded without the LM, MPL achieved similar results to those obtained from PL on both the *clean* and *other* sets (L1 vs. L2). This is different from our previous observations in Table VII, where PL performed better than MPL by using pseudo-labels generated with the LM. With the better seed model trained on more labeled data, the LM was less effective in helping generate pseudo-labels, which is consistent with the findings in [42]. When decoded with the LM, MPL achieved lower WERs on the *other* sets

TABLE XII
WERs [%] AND WRRs [%] ON HIGHER-RESOURCE IN-DOMAIN LS SETTINGS. ALL MODELS WERE TRAINED USING CONFORMER (CFM) ENCODER. ‡ INDICATES TRAINED FOR 100 EPOCHS.

			Decoding without LM						Decoding with LM					
			Dev WER		**Test WER**		**Test WRR**		**Dev WER**		**Test WER**		**Test WRR**	
Setting	**Method**	**Init.**	clean	other	clean	other	clean	other	clean	other	clean	other	clean	other
LS-460	S4 seed (Cfm)	–	4.3	14.0	4.6	13.5	0.0	0.0	2.9	9.4	3.2	9.0	0.0	0.0
LS-460 / LS-500	L1 PL	S4	**3.6**	10.1	**3.8**	10.0	**69.4**	76.3	2.7	7.2	2.9	7.4	52.6	57.3
	L2 MPL	S4	**3.6**	10.1	4.0	9.8	52.3	80.5	**2.6**	6.8	3.0	7.0	37.9	70.9
	L3 MPL‡	L1@ep100	**3.6**	**9.8**	3.9	**9.6**	62.8	**85.3**	**2.6**	**6.6**	**2.8**	**6.9**	**66.2**	**75.9**
	L4 oracle	S4	3.3	8.9	3.5	8.9	100.0	100.0	2.4	6.0	2.6	6.2	100.0	100.0

TABLE XIII
WERs [%] AND WRRs [%] ON HIGHER-RESOURCE OUT-OF-DOMAIN TED3 SETTING. ALL MODELS WERE TRAINED USING CONFORMER (CFM) ENCODER.

			Decoding without LM			Decoding with LM		
Setting	**Method**	**Init.**	**Dev WER**	**Test WER**	**Test WRR**	**Dev WER**	**Test WER**	**Test WRR**
LS-460	S4 seed (Cfm)	–	17.7	18.2	0.0	14.7	15.0	0.0
LS-460 / TED3	M1 PL	S4	13.1	13.1	63.9	12.2	12.8	34.8
	M2 MPL	S4	11.8	11.2	86.6	9.9	**9.5**	83.7
	M3 MPL‡	M1@ep100	**11.6**	**10.9**	**91.0**	**9.8**	**9.5**	**84.7**
	M4 oracle	S4	9.8	10.2	100.0	8.6	8.5	100.0

than PL while keeping WRRs high. Overall, the PL-based initialization (L3) resulted in the best result, but it was less effective on the *clean* sets compared to those in the LS-100 settings (Table VII).

2) Out-of-domain setting: Table XIII shows results on the out-of-domain setting. The seed model trained on LS-460 was also effective for the TED3 setting, achieving much lower WERs than the models trained on less data (S2,S3 vs. S4). The overall trend was consistent with what we observed in Tables III, VIII, and XI, with MPL achieving higher WRRs.

VII. CONCLUSION AND FUTURE WORKS

We proposed momentum pseudo-labeling (MPL), a semi-supervised learning framework for end-to-end ASR. MPL consists of a pair of online and offline models that interact and learn from each other. The online model is trained to predict pseudo-labels generated by the offline model. The offline model maintains an exponential moving average of the online model weights. The interaction between the two models continuously improves the quality of pseudo-labels and permits stabilizing ASR training on unlabeled data. We applied MPL to a CTC-based end-to-end ASR model and conducted experiments on various semi-supervised settings based on LibriSpeech, Libri-Light, and TEDLIUM3. The results demonstrated that MPL significantly improves the seed model and is robust against variations in the amount of labeled/unlabeled data, variations in domain mismatch severity, and over-fitting to LM knowledge. With additional enhancements, e.g., Conformer with group normalization and integration of LM knowledge via IPL, MPL achieved superior performance compared to other pseudo-labeling-based approaches.

Future work should consider introducing filtering [35], [41], [43] as well as other data augmentation techniques [87], [88] into the MPL framework. We also plan to apply MPL to other sequence-to-sequence architectures, such as the attention models [4], [5] and Transducers [7]. Combining MPL with recent powerful pre-trained acoustic models [31], [89] can be another promising direction.

REFERENCES

[1] G. Hinton, L. Deng, D. Yu, G. E. Dahl, A. Mohamed, N. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, T. N. Sainath *et al.*, "Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups," *IEEE Signal Process. Mag.*, vol. 29, no. 6, 2012.

[2] A. Graves, A. Mohamed, and G. Hinton, "Speech recognition with deep recurrent neural networks," in *Proc. ICASSP*, 2013, pp. 6645–6649.

[3] A. Graves and N. Jaitly, "Towards end-to-end speech recognition with recurrent neural networks," in *Proc. ICML*, 2014, pp. 1764–1772.

[4] J. K. Chorowski, D. Bahdanau, D. Serdyuk, K. Cho, and Y. Bengio, "Attention-based models for speech recognition," in *Proc. NeurIPS*, 2015, pp. 577–585.

[5] W. Chan, N. Jaitly, Q. Le, and O. Vinyals, "Listen, attend and spell: A neural network for large vocabulary conversational speech recognition," in *Proc. ICASSP*, 2016, pp. 4960–4964.

[6] 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.

[7] A. Graves, "Sequence transduction with recurrent neural networks," *arXiv preprint arXiv:1211.3711*, 2012.

[8] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to sequence learning with neural networks," in *Proc. NeurIPS*, 2014, pp. 3104–3112.

[9] D. Bahdanau *et al.*, "Neural machine translation by jointly learning to align and translate," in *Proc. ICLR*, 2014.

[10] L. Dong, S. Xu, and B. Xu, "Speech-Transformer: A no-recurrence sequence-to-sequence model for speech recognition," in *Proc. ICASSP*, 2018, pp. 5884–5888.

[11] S. Kriman, S. Beliaev, B. Ginsburg, J. Huang, O. Kuchaiev, V. Lavrukhin, R. Leary, J. Li, and Y. Zhang, "QuartzNet: Deep automatic speech recognition with 1D time-channel separable convolutions," in *Proc. ICASSP*, 2020, pp. 6124–6128.

[12] A. Gulati, J. Qin, C.-C. Chiu, N. Parmar, Y. Zhang, J. Yu, W. Han, S. Wang, Z. Zhang, Y. Wu, and R. Pang, "Conformer: Convolution-augmented Transformer for speech recognition," in *Proc. Interspeech*, 2020, pp. 5036–5040.

[13] C.-C. Chiu, T. N. Sainath, Y. Wu, R. Prabhavalkar, P. Nguyen, Z. Chen, A. Kannan, R. J. Weiss, K. Rao, E. Gonina *et al.*, "State-of-the-art speech recognition with sequence-to-sequence models," in *Proc. ICASSP*, 2018, pp. 4774–4778.

[14] C. Lüscher, E. Beck, K. Irie, M. Kitza, W. Michel, A. Zeyer, R. Schlüter, and H. Ney, "RWTH ASR systems for LibriSpeech: Hybrid vs attention," in *Proc. Interspeech*, 2019, pp. 231–235.

[15] S. Karita, N. Chen, T. Hayashi, T. Hori, H. Inaguma, Z. Jiang, M. Someki, N. E. Y. Soplin, R. Yamamoto, X. Wang *et al.*, "A comparative study on Transformer vs RNN in speech applications," in *Proc. ASRU*, 2019, pp. 449–456.

[16] J. Kahn, M. Rivière, W. Zheng, E. Kharitonov, Q. Xu, P.-E. Mazaré, J. Karadayi, V. Liptchinsky, R. Collobert, C. Fuegen *et al.*, "Libri-Light: A benchmark for ASR with limited or no supervision," in *Proc. ICASSP*, 2020, pp. 7669–7673.

[17] O. Chapelle, B. Scholkopf, and A. Zien, "Semi-supervised learning," *IEEE Trans. Neural Netw.*, vol. 20, no. 3, pp. 542–542, 2009.

[18] A. Tjandra, S. Sakti, and S. Nakamura, "Listening while speaking: Speech chain by deep learning," in *Proc. ASRU*, 2017, pp. 301–308.

[19] T. Hori, R. Astudillo, T. Hayashi, Y. Zhang, S. Watanabe, and J. Le Roux, "Cycle-consistency training for end-to-end speech recognition," in *Proc. ICASSP*, 2019, pp. 6271–6275.

[20] M. K. Baskar, L. Burget, S. Watanabe, R. F. Astudillo *et al.*, "EAT: Enhanced ASR-TTS for self-supervised speech recognition," in *Proc. ICASSP*, 2021, pp. 6753–6757.

[21] S. Karita, S. Watanabe, T. Iwata, A. Ogawa, and M. Delcroix, "Semi-supervised end-to-end speech recognition," in *Proc. Interspeech*, 2018, pp. 2–6.

[22] J. Drexler and J. Glass, "Combining end-to-end and adversarial training for low-resource speech recognition," in *Proc. SLT*, 2018.

[23] Y. Ren, X. Tan, T. Qin, S. Zhao, Z. Zhao, and T.-Y. Liu, "Almost unsupervised text to speech and automatic speech recognition," in *Proc. ICML*, 2019.

[24] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of deep bidirectional transformers for language understanding," in *Proc. NAACL-HLT*, 2019, pp. 4171–4186.

[25] A. Baevski, M. Auli, and A. Mohamed, "Effectiveness of self-supervised pre-training for speech recognition," *arXiv preprint arXiv:1911.03912*, 2019.

[26] S. Ling, Y. Liu, J. Salazar, and K. Kirchhoff, "Deep contextualized acoustic representations for semi-supervised speech recognition," in *Proc. ICASSP*, 2020, pp. 6429–6433.

[27] T. Chen, S. Kornblith, M. Norouzi, and G. Hinton, "A simple framework for contrastive learning of visual representations," in *Proc. ICML*, 2020, pp. 1597–1607.

[28] S. Schneider, A. Baevski, R. Collobert, and M. Auli, "wav2vec: Unsupervised pre-training for speech recognition," in *Proc. Interspeech*, 2019, pp. 3465–3469.

[29] Y.-A. Chung and J. Glass, "Generative pre-training for speech with autoregressive predictive coding," in *Proc. ICASSP*, 2020, pp. 3497–3501.

[30] A. Baevski, S. Schneider, and M. Auli, "vq-wav2vec: Self-supervised learning of discrete speech representations," in *Proc. ICLR*, 2019.

[31] W.-N. Hsu, Y.-H. H. Tsai, B. Bolte, R. Salakhutdinov, and A. Mohamed, "HuBERT: How much can a bad teacher benefit ASR pre-training?" in *Proc. ICASSP*, 2021, pp. 6533–6537.

[32] H. Scudder, "Probability of error of some adaptive pattern-recognition machines," *IEEE Trans. Inf. Theory*, vol. 11, no. 3, 1965.

[33] D.-H. Lee, "Pseudo-label: The simple and efficient semi-supervised learning method for deep neural networks," in *Proc. ICML*, 2013.

[34] B. Li, T. N. Sainath, R. Pang, and Z. Wu, "Semi-supervised training for end-to-end models via weak distillation," in *Proc. ICASSP*, 2019, pp. 2837–2841.

[35] J. Kahn, A. Lee, and A. Hannun, "Self-training for end-to-end speech recognition," in *Proc. ICASSP*, 2020, pp. 7084–7088.

[36] R. Masumura, M. Ihori, A. Takashima, T. Moriya, A. Ando, and Y. Shinohara, "Sequence-level consistency training for semi-supervised end-to-end automatic speech recognition," in *Proc. ICASSP*, 2020, pp. 7054–7058.

[37] F. Weninger, F. Mana, R. Gemello, J. Andrés-Ferrer, and P. Zhan, "Semi-supervised learning with data augmentation for end-to-end ASR," in *Proc. Interspeech*, 2020.

[38] W.-N. Hsu, A. Lee, G. Synnaeve, and A. Hannun, "Semi-supervised speech recognition via local prior matching," *arXiv preprint arXiv:2002.10336*, 2020.

[39] Q. Xu, T. Likhomanenko, J. Kahn, A. Hannun, G. Synnaeve, and R. Collobert, "Iterative pseudo-labeling for speech recognition," in *Proc. Interspeech*, 2020, pp. 1006–1010.

[40] Y. Chen, W. Wang, and C. Wang, "Semi-supervised ASR by end-to-end self-training," in *Proc. Interspeech*, 2020, pp. 2787–2791.

[41] D. S. Park, Y. Zhang, Y. Jia, W. Han, C.-C. Chiu, B. Li, Y. Wu, and Q. V. Le, "Improved noisy student training for automatic speech recognition," in *Proc. Interspeech*, 2020, pp. 2817–2821.

[42] T. Likhomanenko, Q. Xu, J. Kahn, G. Synnaeve, and R. Collobert, "slimIPL: Language-model-free iterative pseudo-labeling," in *Proc. Interspeech*, 2021, pp. 741–745.

[43] S. Khurana, N. Moritz, T. Hori, and J. Le Roux, "Unsupervised domain adaptation for speech recognition via uncertainty driven self-training," in *Proc. ICASSP*, 2021, pp. 6553–6557.

[44] N. Moritz, T. Hori, and J. Le Roux, "Semi-supervised speech recognition via graph-based temporal classification," in *Proc. ICASSP*, 2021, pp. 6548–6552.

[45] S. Laine and T. Aila, "Temporal ensembling for semi-supervised learning," in *Proc. ICLR*, 2017.

[46] A. Tarvainen and H. Valpola, "Mean teachers are better role models: Weight-averaged consistency targets improve semi-supervised deep learning results," in *Proc. NeurIPS*, 2017, pp. 1195–1204.

[47] Y. Higuchi, N. Moritz, J. Le Roux, and T. Hori, "Momentum pseudo-labeling for semi-supervised speech recognition," in *Proc. Interspeech*, 2021, pp. 726–730.

[48] ——, "Advancing momentum pseudo-labeling with Conformer and initialization strategy," *arXiv preprint arXiv:2110.04948*, 2021.

[49] K. He, H. Fan, Y. Wu, S. Xie, and R. Girshick, "Momentum contrast for unsupervised visual representation learning," in *Proc. CVPR*, 2020, pp. 9729–9738.

[50] J.-B. Grill, F. Strub, F. Altché, C. Tallec, P. Richemond, E. Buchatskaya, C. Doersch, B. Avila Pires, Z. Guo, M. Gheshlaghi Azar, B. Piot, k. kavukcuoglu, R. Munos, and M. Valko, "Bootstrap your own latent - A new approach to self-supervised learning," in *Proc. NeurIPS*, 2020, pp. 21 271–21 284.

[51] G. Hinton, O. Vinyals, and J. Dean, "Distilling the knowledge in a neural network," *arXiv preprint arXiv:1503.02531*, 2015.

[52] R. Takashima, S. Li, and H. Kawai, "An investigation of a knowledge distillation method for CTC acoustic models," in *Proc. ICASSP*, 2018, pp. 5809–5813.

[53] G. Kurata and K. Audhkhasi, "Guiding CTC posterior spike timings for improved posterior fusion and knowledge distillation," in *Proc. Interspeech*, 2019, pp. 1616–1620.

[54] J. W. Yoon, H. Lee, H. Y. Kim, W. I. Cho, and N. S. Kim, "TutorNet: Towards flexible knowledge distillation for end-to-end speech recognition," *IEEE Trans. Audio, Speech, Lang. Process.*, vol. 29, pp. 1626–1638, 2021.

[55] D. S. Park, W. Chan, Y. Zhang, C.-C. Chiu, B. Zoph, E. D. Cubuk, and Q. V. Le, "SpecAugment: A simple data augmentation method for automatic speech recognition," in *Proc. Interspeech*, 2019, pp. 2613–2617.

[56] Q. Xie, Z. Dai, E. Hovy, M.-T. Luong, and Q. V. Le, "Unsupervised data augmentation for consistency training," in *Proc. NeurIPS*, 2020.

[57] J. He, J. Gu, J. Shen, and M. Ranzato, "Revisiting self-training for neural sequence generation," in *Proc. ICLR*, 2019.

[58] Q. Xie, M.-T. Luong, E. Hovy, and Q. V. Le, "Self-training with noisy student improves imagenet classification," in *Proc. CVPR*, 2020, pp. 10 687–10 698.

[59] V. Manohar, T. Likhomanenko, Q. Xu, W.-N. Hsu, R. Collobert, Y. Saraf, G. Zweig, and A. Mohamed, "Kaizen: Continuously improving teacher using exponential moving average for semi-supervised speech recognition," in *Proc. ASRU*, 2021, pp. 518–525.

[60] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," in *Proc. NeurIPS*, 2017, pp. 5998–6008.

[61] T. Hori, S. Watanabe, Y. Zhang, and W. Chan, "Advances in joint CTC-attention based end-to-end speech recognition with a deep CNN encoder and RNN-LM," in *Proc. Interspeech*, 2017, pp. 949–953.

[62] P. Guo, F. Boyer, X. Chang, T. Hayashi, Y. Higuchi, H. Inaguma, N. Kamo, C. Li, D. Garcia-Romero, J. Shi *et al.*, "Recent developments on ESPnet toolkit boosted by Conformer," in *Proc. ICASSP*, 2021, pp. 5874–5878.

[63] F. Chollet, "Xception: Deep learning with depthwise separable convolutions," in *Proc. CVPR*, 2017, pp. 1251–1258.

[64] Z. Dai, Z. Yang, Y. Yang, J. G. Carbonell, Q. Le, and R. Salakhutdinov, "Transformer-XL: Attentive language models beyond a fixed-length context," in *Proc. ACL*, 2019, pp. 2978–2988.

[65] Y. Lu, Z. Li, D. He, Z. Sun, B. Dong, T. Qin, L. Wang, and T.-Y. Liu, "Understanding and improving Transformer from a multi-particle dynamic system point of view," *arXiv preprint arXiv:1906.02762*, 2019.

[66] J. Chorowski and N. Jaitly, "Towards better decoding and language model integration in sequence to sequence models," *arXiv preprint arXiv:1612.02695*, 2016.

[67] Y. Higuchi, N. Chen, Y. Fujita, H. Inaguma, T. Komatsu, J. Lee, J. Nozaki, T. Wang, and S. Watanabe, "A comparative study on non-autoregressive modelings for speech-to-text generation," in *Proc. ASRU*, 2021, pp. 47–54.

[68] B. Li, A. Gulati, J. Yu, T. N. Sainath, C.-C. Chiu, A. Narayanan, S.-Y. Chang, R. Pang, Y. He, J. Qin *et al.*, "A better and faster end-to-end model for streaming ASR," in *Proc. ICASSP*, 2021, pp. 5634–5638.

[69] Y. C. Liu, E. Han, C. Lee, and A. Stolcke, "End-to-end neural diarization: From Transformer to Conformer," in *Proc. Interspeech*, 2021, pp. 3081–3085.

[70] J. Kim, J. Lee, and Y. Lee, "Generalizing RNN-transducer to out-domain audio via sparse self-attention layers," *arXiv preprint arXiv:2108.10752*, 2021.

[71] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *Proc. ICML*, 2015, pp. 448–456.

[72] S. Ioffe, "Batch renormalization: Towards reducing minibatch dependence in batch-normalized models," in *Proc. NeurIPS*, 2017, pp. 1942–1950.

[73] Y. Wu and K. He, "Group normalization," in *Proc. ECCV*, 2018, pp. 3–19.

[74] V. Panayotov, G. Chen, D. Povey, and S. Khudanpur, "Librispeech: An ASR corpus based on public domain audio books," in *Proc. ICASSP*, 2015, pp. 5206–5210.

[75] F. Hernandez, V. Nguyen, S. Ghannay, N. Tomashenko, and Y. Estève, "TED-LIUM 3: Twice as much data and corpus repartition for experiments on speaker adaptation," in *Proc. SPECOM*, 2018, pp. 198–208.

[76] D. Povey, A. Ghoshal, G. Boulianne, L. Burget, O. Glembek, N. Goel, M. Hannemann, P. Motlicek, Y. Qian, P. Schwarz *et al.*, "The Kaldi speech recognition toolkit," in *Proc. ASRU*, 2011.

[77] T. Kudo, "Subword regularization: Improving neural network translation models with multiple subword candidates," in *Proc. ACL*, 2018, pp. 66–75.

[78] S. Watanabe, T. Hori, S. Karita, T. Hayashi, J. Nishitoba, Y. Unno, N. Enrique Yalta Soplin, J. Heymann, M. Wiesner, N. Chen, A. Renduchintala, and T. Ochiai, "ESPnet: End-to-end speech processing toolkit," in *Proc. Interspeech*, 2018, pp. 2207–2211.

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

[80] A. Y. Hannun, A. L. Maas, D. Jurafsky, and A. Y. Ng, "First-pass large vocabulary continuous speech recognition using bi-directional recurrent DNNs," *arXiv preprint arXiv:1408.2873*, 2014.

[81] N. Moritz, T. Hori, and J. Le Roux, "Streaming end-to-end speech recognition with joint CTC-attention based models," in *Proc. ASRU*, 2019, pp. 936–943.

[82] J. Ma and R. Schwartz, "Unsupervised versus supervised training of acoustic models," in *Proc. Interspeech*, 2008, pp. 2374–2377.

[83] Y. Zhang, J. Qin, D. S. Park, W. Han, C.-C. Chiu, R. Pang, Q. V. Le, and Y. Wu, "Pushing the limits of semi-supervised learning for automatic speech recognition," *arXiv preprint arXiv:2010.10504*, 2020.

[84] D. Ulyanov, A. Vedaldi, and V. Lempitsky, "Instance normalization: The missing ingredient for fast stylization," *arXiv preprint arXiv:1607.08022*, 2016.

[85] J. L. Ba, J. R. Kiros, and G. E. Hinton, "Layer normalization," *arXiv preprint arXiv:1607.06450*, 2016.

[86] I. A. Sheikh, E. Vincent, and I. Illina, "Transformer versus LSTM language models trained on uncertain ASR hypotheses in limited data scenarios," 2021, working paper or preprint. [Online]. Available: https://hal.inria.fr/hal-03362828

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

[88] Z. Tüske, G. Saon, K. Audhkhasi, and B. Kingsbury, "Single headed attention based sequence-to-sequence model for state-of-the-art results on switchboard," *Proc. Interspeech*, pp. 551–555, 2020.

[89] 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, pp. 12 449–12 460.

Yosuke Higuchi received the B.E. and M.E. degrees in computer science and engineering from Waseda University, Tokyo, Japan, in 2019 and 2021, respectively. He is currently working toward the Ph.D. degree at Waseda University.

Niko Moritz received the B.Sc. degree in imaging physics from the University of Applied Science Bremen, Germany, and the M.Sc. degree in engineering physics as well as the Ph.D. degree from the University of Oldenburg, Germany, in 2016. From 2009 to 2018, he was working on automatic speech recognition in the Project Group Hearing, Speech, and Audio Technology at the Fraunhofer Institute for Digital Media Technology, Oldenburg, Germany. He joined Mitsubishi Electric Research Laboratories (MERL), Cambridge, USA, in 2018 as a Research Scientist. Since 2021 he is a Research Scientist at Meta AI, London, UK.

Jonathan Le Roux (Senior Member, IEEE) received the B.Sc. and M.Sc. degrees in mathematics from the École Normale Supérieure, Paris, France, and the Ph.D. degree from the University of Tokyo, Tokyo, Japan, and Université Pierre et Marie Curie, Paris, France. He is currently a Senior Principal Research Scientist and the Speech and Audio Senior Team Leader with Mitsubishi Electric Research Laboratories, Cambridge, MA, USA. From 2009 to 2011, he was a Postdoctoral Researcher with NTT's Communication Science Laboratories. He has contributed to more than 120 peer-reviewed papers and 20 granted patents in his research field, which includes signal processing and machine learning applied to speech and audio. He is the Founder and Chair of the Speech and Audio in the Northeast (SANE) series of workshops.

Takaaki Hori (SM'14) received the B.E. and M.E. degrees in electrical and information engineering, and the Ph.D. degree in system and information engineering from Yamagata University, Yonezawa, Japan, in 1994, 1996, and 1999, respectively. From 1999 to 2015, he had been engaged in researches on speech recognition and spoken language processing at Cyber Space Laboratories and Communication Science Laboratories in Nippon Telegraph and Telephone (NTT) Corporation, Japan. In 2015, he joined Mitsubishi Electric Research Laboratories (MERL), Cambridge, MA, USA as a Principal Research Scientist. Since 2022, he is a Machine Learning Researcher at Apple, Inc. He has authored more than 100 peer-reviewed papers in speech and language research fields. He currently serves as a member of the IEEE Speech and Language Processing Technical Committee.