TERA: Self-Supervised Learning of Transformer Encoder Representation for Speech

Andy T. Liu, Shang-Wen Li, and Hung-yi Lee

***Abstract*—We introduce a self-supervised speech pre-training method called TERA, which stands for Transformer Encoder Representations from Alteration. Recent approaches often learn by using a single auxiliary task like contrastive prediction, autoregressive prediction, or masked reconstruction. Unlike previous methods, we use alteration along three orthogonal axes to pre-train Transformer Encoders on a large amount of unlabeled speech. The model learns through the reconstruction of acoustic frames from their altered counterpart, where we use a stochastic policy to alter along various dimensions: time, frequency, and magnitude. TERA can be used for speech representations extraction or fine-tuning with downstream models. We evaluate TERA on several downstream tasks, including phoneme classification, keyword spotting, speaker recognition, and speech recognition. We present a large-scale comparison of various self-supervised models. TERA achieves strong performance in the comparison by improving upon surface features and outperforming previous models. In our experiments, we study the effect of applying different alteration techniques, pre-training on more data, and pre-training on various features. We analyze different model sizes and find that smaller models are strong representation learners than larger models, while larger models are more effective for downstream fine-tuning than smaller models. Furthermore, we show the proposed method is transferable to downstream datasets not used in pre-training.**

***Index Terms*—Self-supervised, pre-training, representation.**

I. Introduction

UNLIKE humans, capable of self-learning through experiences and interactions, current real-world speech applications like automatic speech recognition (ASR) rely heavily on large amounts of human annotations. For the next generation of speech processing systems to exhibit similar cognitive intelligence levels as humans, machines should be designed to learn from unlabeled data as humans do. In the era of big data, self-supervised learning has emerged as an attractive approach to leverage knowledge from many unlabeled data, and are shown effective for improving downstream systems [1]–[27].

Manuscript received July 12, 2020; revised February 25, 2021 and June 1, 2021; accepted June 29, 2021. Date of publication July 8, 2021; date of current version July 30, 2021. This work was supported in part by the National Science Council, Taiwan, under Contract 110-2628-E-002-001. The work of A. Liu was supported in part by the Frontier Speech Technology Scholarship of National Taiwan University and in part by ASUS AICS. The associate editor coordinating the review of this manuscript and approving it for publication was Dr. Ozlem Kalinli. *(Corresponding author: Hung-yi Lee.)*

Andy T. Liu and Hung-yi Lee are with the Graduate Institute of Communication Engineering, College of Electrical Engineering and Computer Science, National Taiwan University, Taipei 10617, Taiwan (e-mail: f07942089@ntu.edu.tw; hungyilee@ntu.edu.tw).

Shang-Wen Li is with Amazon AI, New York, NY 10001 USA (e-mail: swdanielli@gmail.com).

Digital Object Identifier 10.1109/TASLP.2021.3095662

In self-supervised learning, an auxiliary task (or pre-training task) is formulated, and models are trained to solve it. While solving the auxiliary task, the network is learning a function that maps input to desired representations. Hence the fundamental tenet of self-supervised learning is the design of an auxiliary task, which allows the model to leverage knowledge from unlabeled data. As such, the formulation of the auxiliary task should be carefully chosen. The task should be challenging enough for the model to learn high-level semantic properties and not be too amiable to exploit low-level shortcuts.

After self-supervised pre-training, learned models could be applied to downstream Speech and Language Processing (SLP) tasks through feature-based *speech representation* extraction, or *fine-tuning* as part of the downstream model. Speech representations are compact vectors which aim to capture high-level semantic information from raw speech [1]–[3], [6]–[21], [27]. Thus, the goal of *speech representation* learning is to find a transform that maps the input acoustic features into such vectors. When the pre-trained networks are re-used as features, it provides a useful speech representation to reduce classifier complexity, makes high-level information more accessible, and ultimately improves downstream SLP tasks. Besides, speech representations also help transfer learning and adaptation across different data distributions [6], [7], [9], [20], [21]. On the other hand, the *fine-tuning* approach uses the pre-trained model to initialize a downstream model for supervised training. The parameters of self-supervised learned models are found to be good initialization for ASR [4], [5], [20]–[26].

In this work, we propose TERA: Transformer Encoder Representations from Alteration, where we use alteration on data to pre-train Transformer Encoders [28]. We introduce a total of three types of alteration to form the self-supervised pre-training scheme: 1) time alteration: reconstructing from corrupted blocks of time steps. 2) frequency alteration: reconstructing from missing blocks of frequency bins. 3) magnitude alteration: reconstructing from altered feature magnitudes. These alterations can be applied together or separately in the pre-training process. We apply alteration on data by dynamically sampling through a probabilistic policy to create random alterations. The model acquires information about the content around the corrupted or altered portions, and by reconstructing them, the model learns a more contextualized representation. We illustrated the framework in Fig. 1.

We use the following downstream tasks to evaluate TERA: phoneme classification, keyword spotting, speaker recognition, and automatic speech recognition (ASR). Also, we compare

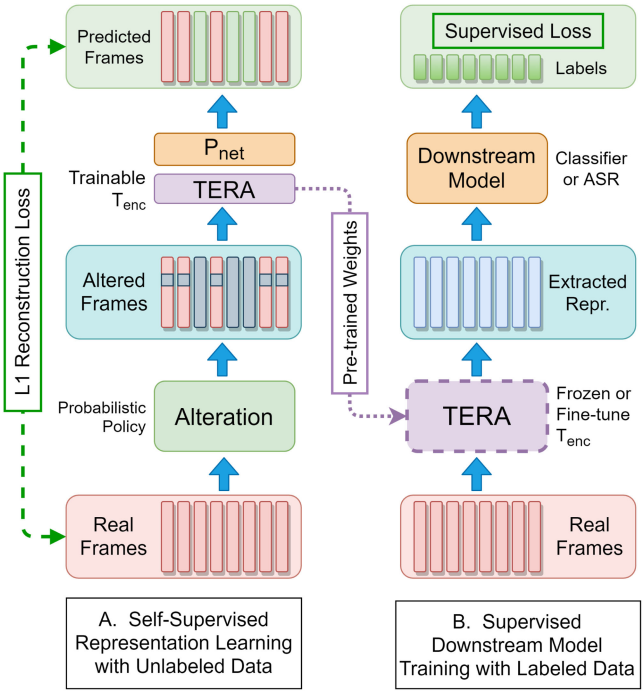

Fig. 1. The illustration of the proposed TERA self-supervised speech representation approach.

the effectiveness of each alteration method separately and in combination. As a result, we confirm that each of the proposed alteration methods guides the model to learn a distinct aspect of speech: 1) The time alteration effectively enforces a more accurate phoneme prediction, keyword detection, and speech recognition, as it leads the model to learn richer phonetic content. 2) The frequency alteration effectively improves speaker prediction accuracy, as it leads the model to encode speaker identity. 3) The magnitude alteration effectively improves performance for all tasks, as it potentially increases data diversity for pre-training.

Different self-supervised frameworks have been widely studied, in Section II we provide a thorough review. Previous work explored mostly for reconstruction on the temporal axis, for example unidirectional (or autoregressive) reconstruction of magnitude or phase from past frames [8]–[10], [15]–[18], [23], or bidirectional reconstruction of a temporal frame from both past and future slices [12]–[14], [20]–[22], [24]–[27]. Our work contrasts with prior work in several ways. Firstly, unlike previous work that only employs reconstruction on the temporal axis, we use reconstruction loss and apply alteration on data along three orthogonal axes, including time, frequency, and magnitude axis. Secondly, most works evaluated their approach with classification tasks only [1], [7], [8], [11], [13]–[17], [20], [21]. In contrast, we moved beyond classification and applied our model to ASR. For a comprehensive investigation, we evaluate our method with four downstream tasks. Thirdly, we explore knowledge transfer between pre-trained models and downstream tasks, an under-investigated problem in speech compared to NLP [29], [30]. We leverage two ways to incorporate the pre-trained model with downstream tasks, where most of the previous work only explored one way of transferring their pre-trained models. Fourthly, we study how self-supervised models behave when pre-trained on a different amount of unlabeled data. Surprisingly, we find that methods that learn from time-only masked reconstruction methods can sometimes not benefit from more unlabeled data. This is because of the reconstruction nature, memorizing all the details, including the unnecessary noise. Additionally, we study the effect of pre-training on various features. We explore four different acoustic features in this work, including log Mel, fMLLR, MFCC, and FBANK. We report results primarily on log Mel and fMLLR. The usage of fMLLR, is not explored before for reconstruction-based methods. Also, none of the previous work explores more than one acoustic feature for their method. Our study finds that using different acoustic features in reconstruction-based learning has a significant effect on pre-trained models and is a parameter choice for researchers. Furthermore, we find smaller pre-trained models perform well for feature extraction over larger models, and larger models tend to be more effective for fine-tuning. Finally, we show explicitly that our approach continues to work well in the face of domain mismatch between pre-training and downstream datasets. For reproducibility of our results, we provide our implementation with pre-trained models and evaluation scripts in the S3PRL Toolkit[1] [31].

II. Related Work

There are two major branches of speech pre-training methods: Contrastive Predictive Coding (CPC) and Reconstruction.

A. Contrastive Losses

1) Contrastive Predictive Coding: The CPC paper [1] describes a form of unidirectional modeling in the feature space, where the model learns to predict the near future frames in an acoustic sequence while contrasting with frames from other sequences or frames from a more distant time. In wav2vec [2], the CPC [1] loss is used to pre-train speech representations for speech recognition, and experiment results show self-supervised pre-training improves supervised speech recognition.

2) CPC With Quantization: In the work of vq-wav2vec [3], the wav2vec [2] approach is incorporated with the well-performing Natural Language Processing (NLP) algorithm – Bidirectional Encoder Representations from Transformers (BERT) [32], [33]. The vq-wav2vec [3] approach learns BERT-like speech representations through a two-stage training pipeline. In a follow-up work of BERT + vq-wav2vec [4], the pre-trained vq-wav2vec [3] model is directly fine-tuned on transcribed speech using a Connectionist Temporal Classification (CTC) [34] loss instead of feeding the representations into a task-specific model. In wav2vec 2.0 [5], the vq-wav2vec method is improved to a single-stage training scheme through time masking in the latent space.

3) Improving CPC: In recent research, the CPC loss has also been extended and applied to bidirectional context networks. Bidirectional CPC (Bidir-CPC) [6] representations are learned

[1][Online]. Available: https://github.com/s3prl/s3prl The S3PRL Toolkit: [Online]. Available: https://github.com/s3prl/s3prl

from bidirectional predictive models. The multi-layer CNN encoder network is shared for both directions, but two autoregressive GRU context networks read encoded observations from the forward and backward contexts. The Modified CPC [7] focuses on improving the CPC approach by introducing several modifications to the original. These modifications include changing the batch normalization to channel-wise normalization, replacing the linear prediction layer with a Transformer layer [28], and replacing the GRU context network with LSTM cells.

In this work, TERA is compared with several contrastive learning methods, including CPC [1], wav2vec [2], vq-wav2vec [3], BERT + vq-wav2vec [4], wav2vec 2.0 [5], Bidir-CPC [6], and Modified CPC [7].

B. Reconstruction Losses

Another recently emerged branch of speech pre-training approach devotes its attention on reconstruction losses.

1) Autoregressive Reconstruction: Primarily inspired by language models (LM) for text, the Autoregressive Predictive Coding (APC) [8], [9] model can be seen as a speech version of LM. The APC approach uses an autoregressive model to encode temporal information of past acoustic sequence; the model then predicts future frames like a recurrent-based LM [35] while conditioning on past frames. In [10], the APC objective is extended to multi-target training. The new objective predicts not only the future frame conditioning on previous context but also past memories. In VQ-APC [11], a vector quantization (VQ) layer is used with the APC objective, which imposes a bottleneck and forces the model to learn better representations. In DeCoAR [12], combining the bidirectionality of ELMo [36] and the reconstruction objective of APC [8], [9], models were able to learn deep contextualized acoustic representations.

2) Time-Only Masked Reconstruction: Largely inspired by the Masked Language Model (MLM) task from BERT [32], [33], [37] and Permutation Language Modeling (PLM) from XL-Net [38], recent work [20]–[27] have explored using BERT-style tasks to pre-train speech encoders. These approaches adopt the NLP pre-training technique to continuous speech. In Mockingjay [20], input frames of speech are masked to zero to pre-train Transformer Encoders. In Audio ALBERT [21], Mockingjay is modified to have shared parameters across Transformer layers. In [39], Mockingjay is shown to be effective in defending adversarial black-box attacks. And in [40], the self-attention of Mockingjay is shown to be meaningful and explainable. On the other hand, TERA can be seen as an improved version of Mockingjay [20]. Using the time alteration alone as pre-training objective reduces TERA to Mockingjay.

In [24], [26], time-only masked reconstruction following the standard BERT masking policy is employed to pre-train ASR encoders. In [25], a simpler masking policy is employed, where input features are divided into chunks of four frames, and masking on chunks are applied with a probability of 15%. In Speech-XLNet [38], models learn by reconstructing from shuffled input speech frame orders rather than masked frames. In [22], SpecAugment [41] is applied on input frames to pre-train ASR encoders (bi-GRUs). In wav2vec 2.0 [5], time masking is applied in the latent space. In Non-autoregressive Predictive Coding (NPC) [27], time masking is introduced through Masked Convolution Blocks, rather than on the input data. In [42], phoneme posterior vectors are used to train a standard BERT [32], [38] model. The phoneme posterior vectors are output from a supervised acoustic model, which requires CTC loss training over the ground-truth phonemes. Also, in [43], CTC loss is used along with time-only masked reconstruction training to learn phonetic representations. As [42] and [43] both use phoneme labels for CTC training, they diverge from other works that are fully self-supervised.

3) Learning From Other Reconstruction Losses: Other than autoregressive and time-only masked reconstruction losses, previous works have also explored the reconstruction of different targets or frameworks, including temporal slice estimation, gap estimation, autoencoders, phase prediction, and Markov Models. In Audio2Vec [13], [14], the model learns through reconstructing a spectrogram slice from past and future slices; this can be seen as a speech version of the NLP Word2Vec [44] variants CBoW (continuous bag-of-words) and skip-gram. The TemporalGap [13], [14] approach learns through estimating the temporal gap between two short audio segments extracted at random from the same audio clip. In [15], speech representations are learned by applying autoencoding neural networks to speech waveform. Apart from reconstructing spectrograms, in [17], representations are learned through reconstructing the phase of the short-time Fourier transform from its magnitude. In PASE [18], a single neural encoder learns to solve multiple self-supervised tasks at once, including reconstruction of waveform, Log power spectrum, MFCC, prosody, and other binary discrimination tasks. The ConvDMM [19] approach learns speech representations with convolutional neural networks and Markov Models. Although The design of the auxiliary task fundamentally decides what the model learns through its reconstruction.

In this work, TERA is compared with several reconstruction learning methods, including APC [8], [9], VQ-APC [11], DeCoAR [12], Mockingjay [20], Audio ALBERT [21], SpecAugment [22], [41], and NPC [27].

III. Proposed Methodology

A. Alteration on Data

As illustrated in Fig. 1 A, the input acoustic frames (outlined in the red box) and target predicted frames (outlined in the green box) could be any acoustic features, such as MFCC, FBANK, fMLLR, or log Mel. We show a sample of 80-dimensional log Mel feature sequence from the LibriSpeech [45] *train-clean-100* subset in Fig. 2 A. We denote the entire speech corpus as $\mathcal{X}$ and the acoustic features of the utterance sampled from $\mathcal{X}$ as $\overrightarrow{x}$. The length (the number of frames) and the height (the number of frequency bins) of $\overrightarrow{x}$ is denoted as L_x and H_x, respectively. Below, we introduce how we use different methods to alter the input $\overrightarrow{x}$.

1) Time Alteration: Our model learns bidirectional representations from past and future contexts by altering contiguous segments along the time axis. In time alteration, a certain percentage of input frames are altered during training, and the model

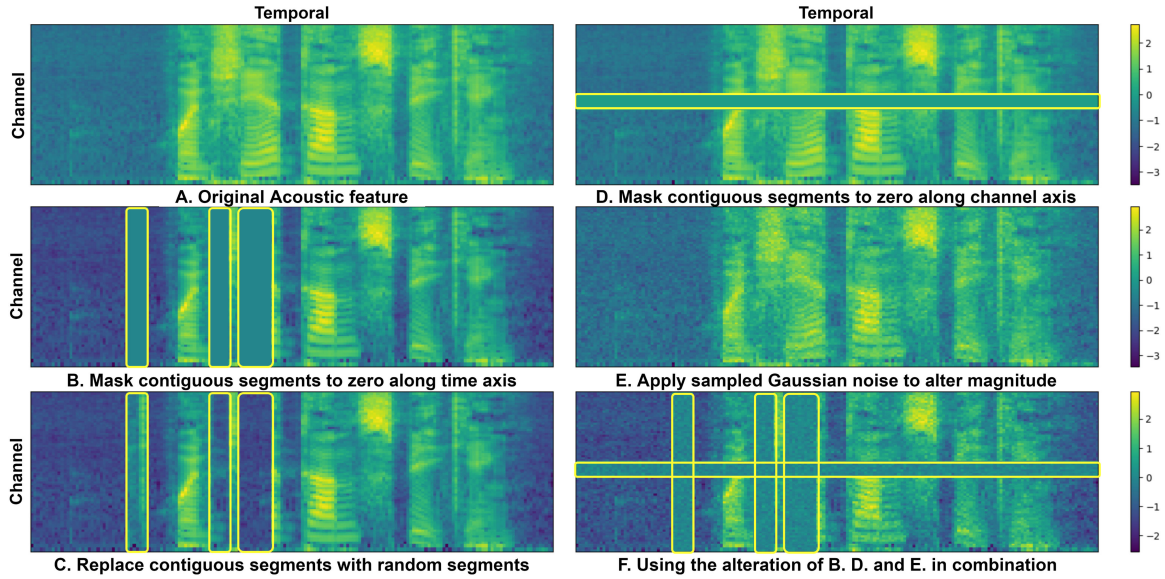

Fig. 2. The illustration of different inputs with various alteration applied for the proposed auxiliary objective. The altered part is highlighted in yellow.

attempts to reconstruct the corrupted span from neighboring frames. We randomly select T_{num} amount of starting indexes I_T without replacement to alter the input utterance. The amount T_{num} is given as the maximum time alteration percentage P_T normalized by the time alteration width W_T:

$$T_{num} = \lfloor P_T \times L_x \div W_T \rceil \tag{1}$$

Note that if time alteration width $W_T = 1$, then $T_{num} = P_T \times L_x$. For each starting index location i_t in I_T, we alter W_T consecutive frames from i_t according to the following stochastic alteration policy: 1) 80% of the time, we mask all the selected frames to zero. 2) 10% of the time, we replace all with random segments of frames. 3) For the rest 10% of the time, we do nothing and leave the frames in $\overrightarrow{x}$ unchanged. The design of Case 3) is to allow the model to receive real inputs during training and addresses the train-test inconsistency problem. This inconsistency problem comes from the absence of alteration during inference time, and the model will only receive acoustic features without alteration.

We illustrate the masking and replacing of frames in Fig. 2 B and 2 C, respectively. Our time alteration policy is more sophisticated than other time-only masked reconstruction approaches [22], [25], where they simply mask a percentage with zeroed-out spans, unlike ours that have random and real frames. We set the time alteration width W_T to 7 frames, which corresponds to 85 ms of speech. The time alteration width W_T then lies in the average phoneme duration range (average phone duration is around 50 ms to 100 ms at usual rates of 10 to 20 phones per second). We set the P_T percentage of total altered frames to 15%, as suggested in [20], [32], [33]. We allow time alteration blocks to overlap each other, hence resulting in the larger highlighted yellow box in the left of Fig. 2 B and 2 C. With overlapping, we generate a longer altered span ($> W_T$) and force the model to infer on more global structure rather than a fixed local span (W_T). The idea behind time alteration is that a model that can predict the partial loss of small speech segments should provide a contextualized understanding of previous and future content. Our experiments show that the proposed time alteration is the essential element that drives models to learn bidirectional understanding, resulting in a substantial improvement compared to models that are not using the time alteration method.

2) Frequency Alteration: Our second pre-training method is frequency alteration. It is largely inspired by SpecAugment proposed for ASR augmentation [41], and the ASR pre-training scheme proposed in [22]. We randomly mask the values of a block of consecutive frequency bins to zero for all time steps across the input sequence for this alteration. The block of masked frequency is selected by first sampling the width of block W_C from $\{0, 1, ..., W_C\}$ uniformly. Then, we sample a frequency index I_C from $\{0, 1, ..., H_x - W_c - 1\}$, where H_x is the number of frequency bins in input sequence $\overrightarrow{x}$. The frequency bins from I_C to $I_C + W_c - 1$ are those to be masked. Note that the policy will mask none of the frequencies for $1/(W_C + 1)$ of the time. Thus, from time to time, the model will receive inputs with all of the frequency information. By allowing the model to receive real inputs again addresses the inconsistency between training and inference time.

We illustrate this alteration's effect on input sequence in Fig. 2 D. Unlike the time alteration case, where we sample many blocks for alteration as visualized in Fig. 2 B and 2 C, we only sample a single block for frequency alteration in each utterance. The reason is that acoustic sequences can be arbitrarily long and temporally smooth [46], while there are only a limited and fixed number of frequencies H_x. Hence, we select multiple blocks for time alteration, but only one block for the frequency axis. Following the work of [22], we set the maximum frequency alteration width W_C to 16 frequency bins (20% of the 80-dimension feature). The intuition behind frequency alteration is that a model that can predict the partial loss of frequency information should learn a high-level understanding along the frequency axis.

As we will show in our experiments, we find that using frequency alteration provides a more linearly spreadable speaker

representation and a stronger speaker recognizer. Surprisingly, encoding speaker information through this objective does not compromise phoneme classification or ASR performance but instead increases performance. This makes TERA not only suitable for tasks that only require speaker information (e.g. speaker recognition) and tasks that only require phonetic information (e.g. speech recognition), but also beneficial for tasks that requires both speaker and phonetic information at the same time (e.g. voice conversion [16]).

3) Magnitude Alteration: We introduce the third method, magnitude alteration, by applying sampled Gaussian noise to augment input sequences' magnitude with a probability P_N. For P_N of the time, we sample a random magnitude matrix $\overrightarrow{z}$ of dimensions L_x and H_x, which has the same shape as $\overrightarrow{x}$. Each element in $\overrightarrow{z}$ is sampled from the normal distribution $\mathcal{N}$ with zero mean and 0.2 variance. We then add $\overrightarrow{z}$ on top of the real frames of $\overrightarrow{x}$. We show the effect of magnitude alteration in Fig. 2 E, where we apply magnitude alteration to the original utterances $\overrightarrow{x}$. By altering input magnitude, we potentially increase the amount of pre-training data (which is similar to the idea of data augmentation [41]). Additionally, magnitude alteration offers another variation to all the 'mask to zero' cases described in Section III-A1 (time alteration) and Section III-A2 (frequency alteration). We illustrate this new 'mask to noise' variation in Fig. 2 F, where the selected blocks of time and frequency are now with random magnitudes instead of zeros. Empirically, altering magnitude provides a performance benefit for all downstream tasks, thanks to the increased input data variety. Also, the gain from magnitude alteration is additive to that of other alteration techniques.

B. Pre-Training TERA

We use the three alteration techniques to formulate the TERA self-supervised pre-training task, where the model is required to minimize the reconstruction error of acoustic features given altered frames as input. The proposed three alteration methods can be used separately or used together as a mixture, as shown in Fig. 2. The complete pre-training process of TERA is described in Algorithm 1, where we show how three alteration on data are deployed through the proposed stochastic policy. The stochastic policy dynamically samples random pattern based on the applied alterations every time we feed an input sequence to the model. We denote the altered input as $\hat{x}$. In the case where multiple alterations are applied on the same input, $\overrightarrow{x}$ is used again to represent carried-on alterations. After input alteration, we feed $\hat{x}$ into the Transformer Encoders T_{enc} and the prediction network P_{net}. The architecture of P_{net} is consist of a 2-layer feed-forward network with a hidden size of 768. For T_{enc}, we use Transformer Encoders with a hidden size of 768, number of self-attention heads as 12, dropout rate of 0.1, and the intermediate feed-forward layer hidden size as 3072. We primarily report results on three model sizes: *base* (Layer=3, Parameters=21.3 M), *medium* (Layer=6, Parameters=42.6 M), and *large* (Layer=12, Parameters=85.1 M). The Transformer Encoders T_{enc} and the prediction network P_{net} are connected to reconstruct $\overrightarrow{x}$ from $\hat{x}$.

Algorithm 1: The TERA Self-Supervised Pre-Training Algorithm

Input: Dataset $\mathcal{X}$, total training steps T_{steps}, time alteration percentage P_T, time alteration width W_T, frequency alteration width W_C, magnitude alteration percentage P_N.
Output: Transformer Encoders parameters θ_{enc}
Initialize Transformer Encoders T_{enc} parameters θ_{tenc}
Initialize Prediction Network P_{net} parameters θ_{pnet}
for $t = 1$ to T_{steps} **do**
  Sample speech utterance $\overrightarrow{x}$ from $\mathcal{X}$
  if apply time alteration **then**
    $L_x =$ number of frames in $\overrightarrow{x}$
    $T_{num} = \lfloor P_T \times L_x \div W_T \rceil$
    $I_T =$ Sample T_{num} amount of starting indexes
    Sample value v from uniform distribution $\mathcal{U}(0, 1)$
    if $v < 0.8$ **then**
      for all i_t in I_T **do**
        $\hat{x} \leftarrow$ mask W_T number of consecutive frames in $\overrightarrow{x}$ to zero
      end for
    else if $v >= 0.8$ and $v < 0.9$ **then**
      for all i_t in I_T **do**
        $\hat{x} \leftarrow$ replace W_T number of consecutive frames in $\overrightarrow{x}$ with randomly sampled consecutive frames from the same utterance
      end for
    else
      Do nothing, $\hat{x} \leftarrow \overrightarrow{x}$
    end if
  end if
  if apply frequency alteration **then**
    if apply time alteration **then**
      Apply on top of previous alteration, $\overrightarrow{x} \leftarrow \hat{x}$
    end if
    $W_c =$ Sample a frequency width from $\mathcal{U}(0, W_C)$
    $H_x =$ number of frequencies in $\overrightarrow{x}$
    $I_C =$ Sample a frequency index from $\mathcal{U}(0, H_x - W_c)$
    $\hat{x} \leftarrow$ starting from I_C, mask W_c consecutive frequency bins in $\overrightarrow{x}$ to zero
  end if
  if apply magnitude alteration **then**
    if apply time or frequency alteration **then**
      Apply on top of previous alterations, $\overrightarrow{x} \leftarrow \hat{x}$
    end if
    Sample value v from uniform distribution $\mathcal{U}(0, 1)$
    if $v < P_N$ **then**
      Sample magnitude vector $\overrightarrow{z}$ from $\mathcal{N}(0, 0.2)$
      Apply noise $\hat{x} \leftarrow \overrightarrow{x} + \overrightarrow{z}$ to alter magnitude
    else
      Do nothing, $\hat{x} \leftarrow \overrightarrow{x}$
    end if
  end if
  Compute L1 reconstruction loss to update:

$$L_{rec}(\theta_{tenc}, \theta_{pnet}) = \| \overrightarrow{x} - P_{net}(T_{enc}(\hat{x})) \|_1 \quad (2)$$

end for

L1 reconstruction loss is then computed between input $\overrightarrow{x}$ and network output from P_{net} to update the network parameters θ_{tenc} and θ_{pnet}. We use gradient descent training with mini-batches of size 32 to find model parameters that minimize the L1 loss under the self-supervised pre-training task. We employ the AdamW optimizer [47] for updating model parameters, where the learning rate is warmed up over the first 7% of total training steps T_{steps} to a peak value of $2e^{-4}$ and then linearly decayed. Our pre-training setup can be accommodated in a single 1080Ti GPU with 11 GB of memory. Computational efficiency allows interested parties to easily train our model with their data without massive computational resources. The models are trained with fixed total training steps T_{steps} (details in Section IV-A). After pre-training, the parameters θ_{tenc} of the Transformer Encoders T_{enc} are retained for downstream tasks, while the prediction network P_{net} is discarded, as illustrated in Figure 1.

In this work, our models' input is an 80-dimensional log Mel spectrogram if not specified otherwise. We also explore pre-training with other acoustic features, including 39-dimensional MFCC, 80-dimensional FBANK, and 40-dimensional fMLLR. We extract all of the features with a window of 25 ms and a stride of 10 ms. We apply per-utterance CMVN (cepstral mean and variance normalization) to the features. We set the total pre-training steps T_{steps} of TERA as 200 k and 1 M for 100 hours and 960 hours of pre-training data, respectively.

C. Incorporating With Downstream Tasks

We investigate different ways to incorporate the learned TERA model to downstream tasks.

1) Representation Extraction: We first use the weighted sum technique to investigate which layer is best for representation extraction. We use a learnable weighted sum to integrate hidden states from all layers of the self-supervised model when training with downstream tasks, similar to the ELMO [36] approach. We then analyze the learned weights and find that for TERA, APC [8], [9], and vq-wav2vec [3], the weighted sum always favors the last layer the most (we investigate with phoneme and speaker classification tasks). Hence in this work, we extract representations from TERA's deepest layer, which is essentially the hidden states of the last Transformer Encoder layer. The extracted representation is fed to downstream classifiers as input and replacing surface features. Parameters of TERA are frozen when training downstream tasks in this approach. In later experiments, we use this approach if not specified otherwise.

2) Fine-Tuning: The second approach is to fine-tune the TERA model with downstream models. Here the output of TERA is connected to a downstream model of any kind, as illustrated in Fig. 1. We then update the pre-trained TERA together with random initialized downstream models. We denote this approach as *fine-tune* to distinguish from representation extraction in later experimental tables and figures.

IV. Experimental Setup

We use four downstream tasks for evaluation: phoneme classification, speaker recognition, keyword spotting, and speech recognition. We use publicly known and available settings for our downstream tasks to link our works with previous ones and allow easier comparison. For all experiments, we train with fixed random seeds for consistency.

A. Datasets

We use a total of three datasets. We consider three subsets of LibriSpeech for the pre-training of TERA: the *train-clean-100*, the *train-clean-360*, and the *train-other-500* subset. The three subsets add up to a total of 960 hours of data. We also use the LibriSpeech [45] dataset for downstream tasks of phone classification, speaker recognition, and speech recognition. The *train-clean-100* subset is used in the phoneme classification and speaker recognition task. For ASR, we use the *train-clean-100* as labeled data, and the *dev-clean* and *test-clean* subset are used for evaluation. We use the TIMIT [48] dataset to evaluate the transferability of pre-trained models. We do not use the TIMIT dataset for pre-training, but we use it for downstream phoneme classification and ASR tasks. We consider two subsets of TIMIT: the training set and the complete test set. As is usually done, we derive the development set and the core test set from the complete test set, according to the Kaldi [49] TIMIT recipe and [50]. We report results by training downstream tasks on the training set and evaluating with the development and core test sets. For keyword spotting, we use the Speech Commands [51] dataset. We consider two subsets of Speech Commands: the training set and the test set. We derive the development set from the validation list provided in Speech Commands [51]. We report results by training keyword spotting models on the training set, and evaluating on the development set and test set. The Speech Commands dataset is also not used for pre-training.

B. Phoneme Classification Setup

1) Phoneme Classification on LibriSpeech: We measure the frame-wise phoneme prediction performance with classifiers trained on top of representations. Following previous work [1], [7], we adapt the common setup using 41 possible phoneme classes and the *train-clean-100* subset of LibriSpeech [45]. To make the comparison as fair as possible, we use aligned phoneme labels and train/test split provided in the CPC [1] and Modified CPC [7] literature. We derive 10% of the data from the training split and use it as the development set. Following the previous work [1], we utilize linear classifiers to measure the linear separability of phonemes. We denote these classifiers as *linear*. Additionally, following the work of Modified CPC [7], we also report results from performing linear classification with a concatenation of 8 windows, which matches the average length of a phoneme. We denote this type of classifier setting as *linear concat*. As not all the information encoded is linearly accessible, in addition to measuring linear separability, we also use classifiers with one single hidden layer, following the same settings as in [1]. We denote such setting as *1-hidden*.

2) Phoneme Classification on TIMIT: For TIMIT, we obtain a phoneme label for each frame from the manual phoneme transcriptions provided in TIMIT. We measure accuracy after mapping the 48 phonemes to the smaller set of 39 phoneme classes, as is customary for TIMIT since [52] (the same applies to ASR

on TIMIT). Following the classifier settings for LibriSpeech, we also use the *linear*, *linear concat*, and *1-hidden* classifiers on TIMIT. For both LibriSpeech and TIMIT, we report phoneme classification accuracy (%) on the test set. We investigate the domain shift issue with this setup, where models are pre-trained on LibriSpeech then used on TIMIT for the downstream task.

C. Keyword Spotting Setup

We evaluate representations on the keyword spotting task. We follow the standard setup [51], where the keyword spotting task is a 12-class classification problem. The test set provides equal numbers of instances for each class; hence class accounts for approximately 8.3%. To probe representations, we did not use complex models for this task, as performance may saturate. Instead, we use a two hidden layer feed-forward classifier, with mean pooling overtime applied just before the output layer. We report keyword classification accuracy (%) on the test set. This setup allows us to study the domain shift issue, where models are pre-trained on LibriSpeech then used on Speech Commands for the detection task.

D. Speaker Classification Setup

We evaluate representations on speaker prediction tasks. Following the common experiment setting in [1], [20], [21], we adopt the LibriSpeech [45] *train-clean-100* subset, which consists of 251 speakers. We use the same train/test split as provided in [1]. We evaluate the pre-trained models with two types of speaker classification tasks, frame-wise linear classification, and utterance-wise linear classification. For frame-wise speaker classification, the classifier predicts the speaker for each input frame. We denote this experiment setting as *speaker frame*. As for utterance-wise classification, we average the representation of each utterance over time. Then the classifier predicts speaker identity conditioning on the averaged vector. We denote this setting as *speaker utter*. For both settings, we employ a simple linear classification model. In [1], [21], they also investigate these two speaker classification settings. In general, the *speaker frame* classification task is more difficult than *speaker utter*, however *speaker utter* is a more common scenario for speaker classification. Also, good results in *speaker frame* always imply good results in *speaker utter*, but not the other way around. Hence we report both *speaker frame* and *speaker utter* for completeness. For both tasks, we report speaker classification accuracy (%) on the test set.

E. Hybrid DNN/HMM ASR Setup

We evaluate the performance of ASR models on top of representations. We employ the Hybrid DNN/HMM ASR modeling implemented with the PyTorch-Kaldi [53] toolkit. We investigate two types of DNN settings, *MLP* and *liGRU*. *MLP* is a simple single-layer multilayer perceptron model. *liGRU* is a 5-layer light-gated recurrent unit followed by 2-layers of fully-connected network. In first-pass decoding, we use a 4-gram language model with a beam-search algorithm. The WER score is computed with the *NIST SCTK* scoring toolkit [53]. For lattice rescoring, we use the Kaldi *ConstArpaLm* format language model. The decoding and rescoring scripts are inherited from the Kaldi *s5* recipe [49].

When we utilize TERA as a representation extractor, we feed the output of TERA to *liGRU* or *MLP* and freeze the parameters of TERA during training. As for fine-tuning TERA, we update the TERA model with *liGRU* or *MLP* as part of the DNN component in the hybrid ASR framework. For our ASR experiments, we pre-train TERA on 40-dimensional fMLLR [54] features instead of 80-dimensional log Mel features since the PyTorch-Kaldi ASR toolkit [53] works best with fMLLR inputs. To highlight the effect of pre-training, we use a limited amount of labeled data, i.e., the *train-clean-100* subset, for supervised ASR training. Hyperparameters are tuned on the *dev-clean* subset, and testing results measured from the *test-clean* subset are reported. We report ASR modeling results of LibriSpeech in terms of WER. In addition to evaluating with LibriSpeech, we also benchmark ASR results with TIMIT, where we measure PER. With this setup, we investigate the domain shift issue, where models are pre-trained using data in the domains different than the downstream tasks.

F. Training Downstream Tasks

For both representation extraction and fine-tuning, we use the AdamW [47] optimizer to update models when training with downstream tasks. We use a learning rate of $2e^{-4}$ with a batch size of 16. When applying representation extraction for ASR tasks, we use the RMSPROP optimizer with a learning rate of $2e^{-4}$. We half the learning rate every epoch if the development set error does not drop more than a threshold of 0.001. We use a batch size of 16, and update for 24 epochs. We use the same setting as above for fine-tuning on ASR, except we use a different learning rate for each TERA model. Learning rate is set to $2e^{-4}$ when we fine-tune *base*, $1e^{-4}$ for *medium*, and $5e^{-5}$ for *large*. Empirically, we find that larger models require a lower learning rate during fine-tuning. Hence, we half the learning rate every time the model depth is doubled; this helps stabilize the entire fine-tuning process.

During downstream fine-tuning, we also apply the SpecAugment [41] LD policy. During SpecAugment, the chosen (and possibly overlapping) time and frequency blocks are zeroed out. We omit the use of time warping as masking provides enough regularization. We find SpecAugment is additive to the proposed pre-training approach, as it delays overfitting and improves the final accuracy numbers in the downstream tasks.

V. Results

In Section V-A, we study the effect of applying different combinations of time, frequency, and magnitude alteration. In Section V-B, we compare TERA with recent self-supervised representation methods on downstream tasks. In Section V-C, we study multiple aspects of the TERA pre-training process, including fine-tuning with downstream models, the amount of unlabeled data, the effect of pre-training on various features, various model sizes, and different masking policies. In Section V-D, we compare TERA with approaches where we train

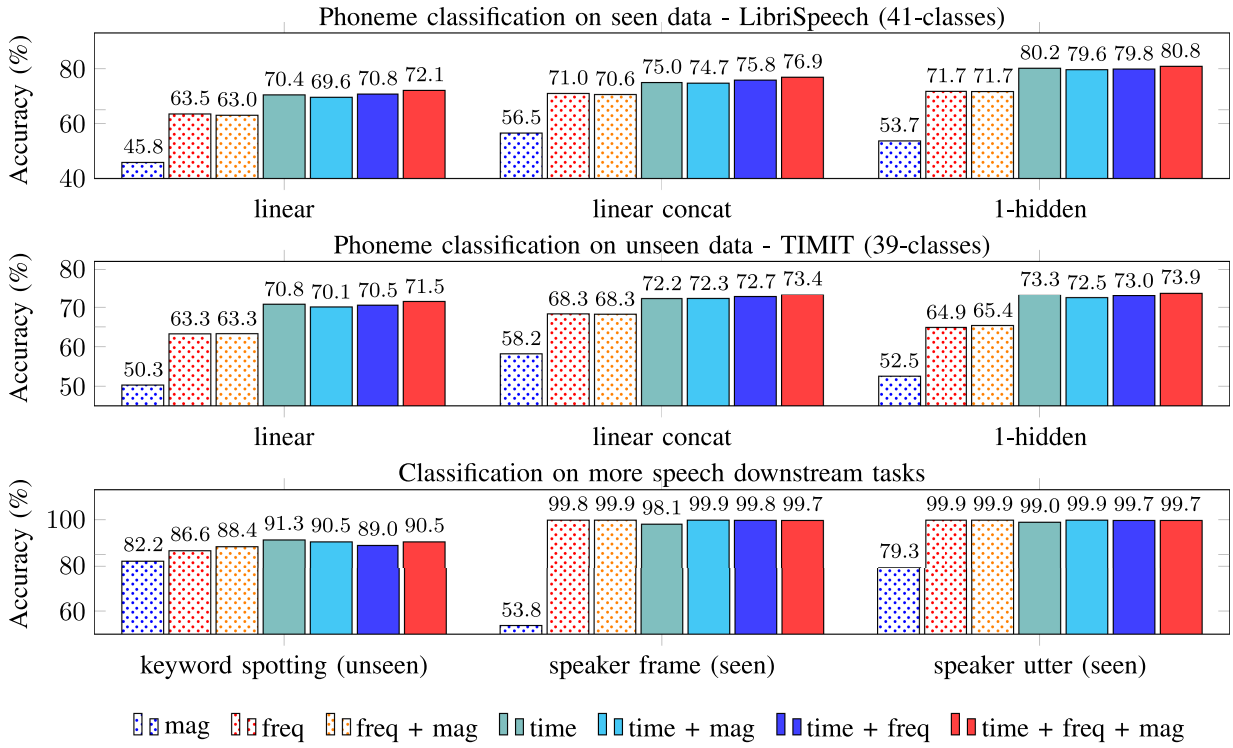

Fig. 3. **The effect of different alterations.** We use phoneme classification, keyword spotting, and speaker recognition on both seen and unseen data to study the effect of different alteration methods. All results are *base* models pre-trained on the LibriSpeech train-clean-100 subset, testing accuracy on different types of classifiers are reported. Methods with unidirectional context (without time alteration) are in dotted color, methods with bidirectional context (with time alteration) are in solid color.

ASR models on frozen speech representations. In Section V-E, TERA is compared with approaches that fine-tune their pre-trained models. In Section V-F, we demonstrate transferring TERA representations from one dataset to another by presenting ASR PER on TIMIT [48].

A. The Effect of Different Alterations

With three types of alterations, there are a total of seven combinations, namely *"time"*, *"freq"*, *"mag"*, *"time+freq"*, *"time+mag"*, *"freq+mag"*, and *"time+freq+mag"*. We pre-train TERA with all seven combinations of alterations on 100 hours of LibriSpeech, then measure the learned representations with downstream tasks. The results are presented in Figure 3. We use TERA for representation extraction (from the last layer), i.e., TERA parameters were frozen when adapting downstream models. We split our methods into two categories: one with the time alteration that allows the model to encode bidirectional information (denoted in a solid color). The second category is the models that don't include the time alteration method (denoted in dotted color). The conclusions made in the following sections are statistically significant, where we measured p-values with statistical significance tests (paired samples t-test or Fisher's exact test). Overall, applying all alterations *"time+freq+mag"* achieves the best performance on average (with an averaged p-value of 0.0492 across nine tasks when compared to *"time"*). Now, we discuss the effect of each alteration in detail.

1) The Effect of Time Alteration: We discuss the effect of time alteration for each downstream task. For phoneme classification, the models with time alteration (in solid color) perform reasonably well, while the models without time alteration (in dotted color) perform relatively poorly. The results of seen data (LibriSpeech) and unseen data (TIMIT) agree with each other and yield similar trends. For keyword spotting, the *"time"* model achieves the best performance. Adding other alterations compromise the performance. The models with time alteration perform better than the models that do not have time alteration. This observation suggests that bidirectional understanding is an essential aspect of detection tasks. For speaker recognition, the existence of time alteration does not compromise the performance. With time alteration alone, the *"time"* model achieves slightly lower but satisfactory performance. From the above discussion, we thus deem the time alteration to be indispensable. We speculate that although Transformer Encoders [28] are bidirectional due to their multi-head self-attention, without the time alteration, the network fails to encode proper context and yield sub-optimal performance. However, through alteration on the time axis, the model establishes a bidirectional understanding of the audio and gives better results.

2) The Effect of Frequency Alteration: For the phoneme classification tasks, adding the frequency alteration during pre-training is effective in boosting performance. The *"time+freq"* and *"time+freq+mag"* model improves over the *"time"* model for both seen and unseen data. For keyword spotting, adding the frequency alteration does not help. For speaker recognition,

TABLE I
DETAILS OF BASELINE FEATURES AND RECENT SPEECH REPRESENTATION APPROACHES. WE EVALUATE THEIR PERFORMANCE ON DOWNSTREAM TASKS AND PRESENT THE RESULTS IN TABLE II

Representation	Network	#Params	Input Feature	Pre-train	Learning Style
MFCC	-	0	13-dim + delta 2	-	-
FBANK	-	0	80-dim + delta 2	-	-
log Mel	-	0	80-dim	-	-
APC [8], [9]	3-GRU	4,064,256	80-dim log Mel	LibriSpeech 360 hr	autoregressive reconstruction
VQ-APC [11]	3-GRU	4,064,256	80-dim log Mel	LibriSpeech 360 hr	autoregressive reconstruction + VQ
CPC [1], [7]	5-Conv 1-Trans	1,843,456	waveform	LibriLight 60k hr	contrastive
vq-wav2vec [3]	20-Conv	6,042,624	waveform	LibriSpeech 960 hr	contrastive + VQ
wav2vec 2.0 [5]	7-Conv 24-Trans	214,658,176	waveform	LibriSpeech 960 hr	contrastive + latent masking
Mockingjay [20]	3-Trans	21,327,360	80-dim log Mel	LibriSpeech 960 hr	time-only masked reconstruction
Audio ALBERT [21]	3-Trans	7,151,616	80-dim log Mel	LibriSpeech 960 hr	time-only masked reconstruction
NPC [27]	3-Conv	19,273,728	80-dim log Mel	LibriSpeech 360 hr	time-only masked reconstruction
TERA (Ours)	3-Trans	21,327,360	80-dim log Mel	LibriSpeech 960 hr	time + freq + mag alteration

we find that the model can achieve strong speaker classification performance by employing the frequency alteration. The *"freq"* model, with frequency alteration alone, is enough to achieve high performance. The *"mag"* and *"time"* models, without frequency alteration, have lower performance. By employing the frequency alteration, models benefit in the phoneme classification and speaker recognition performance, which is surprising and counter-intuitive. It is well known that a robust phonetic system should be speaker invariant [16]. We surmise this phenomenon results from that TERA representations provide more accessibility to phonetic and speaker information and maintain the separability between the two types of information. Downstream models can efficiently learn to extract task-specific information. To sum up this section, the frequency alteration effectively encodes speaker identity while also fine grinding the phonetic quality.

3) The Effect of Magnitude Alteration: For the phoneme classification tasks, adding the magnitude alteration is effective in boosting performance. The *"time+freq+mag"* model improves over the *"time+freq"* model for both seen and unseen data. For keyword spotting, adding the magnitude alteration improves most cases but does not improve over time alteration alone. For speaker recognition, adding the magnitude alteration improves performance for all cases. We observe that by applying the magnitude alteration during pre-training, the model learns to be robust to various inputs. As a result, we improve downstream tasks performance by adding the magnitude alteration.

B. Comparison of Recent Speech Representation Approaches

In this section, we compare TERA with recent speech representation learning methods through evaluating with downstream tasks. We select eight different methods, which cover a wide range of representation learning techniques. We list these methods and their details in Table I. We organize Table I by first grouping methods with similar learning styles, then by their publication date in the order of first to last. We use publicly available pre-trained weights or pre-training scripts provided by their original authors for all of the listed models. Note that all models are pre-trained on LibriSpeech [45] except for CPC [1], [7], which is pre-trained on LibriLight [55]. All of the models and baseline features have a stride of 10 ms, with only one exception where wav2vec 2.0 [5] has a downsample rate of 320, which generates a representation every 20 ms. Interestingly, all of the methods that employ reconstruction loss in their learning objective use an 80-dim log Mel input feature. We also use 80-dim log Mel for consistency and fair comparison. On the other hand, all methods that employ contrastive loss in their learning objective use raw waveform as an input feature. In Table I, we also list the details of baseline features. We present the performance of different representations on downstream tasks in Table II. The pre-trained models are all used for representation extraction (from the last layer), i.e., we freeze parameters when adopting the downstream model and feed representations to the downstream model as input. In the last column, we average the accuracy of all tasks across each row. The above results can be reproduced with the S3PRL toolkit[2], where all the self-supervised models and downstream tasks are available with easy-to-use setups.

1) Phoneme Classification Results: For phoneme classification on seen data (LibriSpeech), TERA outperforms other methods. For phoneme classification on unseen data (TIMIT), TERA outperforms other methods except for CPC on the *linear concat* classifier. Some representations encounter a more significant degradation in performance for unseen data. On the other hand, TERA is not affected by the domain change. In particular, we find that adding the TERA magnitude alteration helps performance for unseen data. As a result, by adding magnitude alteration, we increase accuracy for phoneme classification on TIMIT. We observe that all representations benefit from concatenating neighboring frames, as their performance on *linear concat* is better than *linear*. The *linear concat* classifier provided neighboring information through the concatenated frames. Several representations benefit more from *linear concat* than others, depending on how much temporal information the single feature frame already encodes. We also observe that all representations benefit from a deeper downstream model, as their performance on *1-hidden* is better than *linear*. The *1-hidden* classifier is able to extract more information than linear classifiers. The wav2vec 2.0 method achieves the lowest accuracy for phoneme classification, which contrasts its high performance when used as an initialization for ASR encoder fine-tuning [5]. The under-performing of wav2vec 2.0 here suggests that although large and deep models are suitable for downstream initialization and fine-tuning, they may not be a good choice for feature extraction. Our later experiment shows that TERA-large under-perform TERA-base

TABLE II
PERFORMANCE OF BASELINE FEATURES AND RECENT SPEECH REPRESENTATION APPROACHES. REPRESENTATIONS LISTED IN TABLE I ARE EVALUATED ON THE FOUR DOWNSTREAM TASKS. WE REPORT TESTING ACCURACY (%) AND HIGHLIGHT THE HIGHEST SCORE FOR EACH COLUMN IN BOLD FONT

Representation	Phoneme - LibriSpeech			Phoneme - TIMIT			Keyword	Speaker		Average
	linear	linear concat	1-hidden	linear	linear concat	1-hidden	spotting	frame	utter	
MFCC	47.88	51.46	62.47	54.74	58.55	65.52	87.99	13.18	90.61	59.16
FBANK	48.01	52.16	63.38	55.00	58.35	64.52	86.01	31.35	94.70	61.50
log Mel	41.93	49.94	48.55	47.37	56.04	51.05	72.35	22.38	95.33	53.88
APC [8], [9]	72.76	79.62	78.06	72.90	76.74	74.51	91.04	79.08	**99.63**	80.48
VQ-APC [11]	71.92	79.08	77.62	72.06	75.98	74.11	92.92	68.56	99.21	79.05
CPC [1], [7]	71.31	78.52	77.40	73.10	**77.71**	75.57	94.16	75.47	99.32	80.28
vq-wav2vec [3]	63.38	76.53	66.18	67.52	76.42	68.68	**94.55**	24.76	84.09	69.12
wav2vec 2.0 [5]	56.39	72.72	73.61	59.32	71.76	69.14	68.87	82.53	95.40	72.19
Mockingjay [20]	67.51	72.07	78.70	69.21	71.68	72.53	88.09	97.22	98.35	79.48
Audio ALBERT [21]	68.13	72.62	78.65	69.62	71.37	72.68	90.59	96.65	98.37	79.85
NPC [27]	67.32	76.08	75.35	65.76	72.96	68.42	87.41	27.72	96.16	70.80
TERA time + freq	**74.45**	**78.84**	**82.14**	73.92	75.80	75.98	91.89	**99.57**	99.59	83.58
+ mag	74.07	78.67	81.90	**74.14**	75.92	**76.23**	92.60	99.47	99.48	**83.61**

for feature extraction, but TERA-large outperforms TERA-base for ASR fine-tuning.

2) Keyword Spotting Results: For keyword spotting, all representations achieve a good score. However, wav2vec 2.0 and NPC are not able to surpass the performance of baseline features. The wav2vec 2.0 method achieves the lowest score, which again suggests that pre-trained models may work well for fine-tuning, but it does not always imply that they can generate meaningful representations. The NPC method and Mockingjay achieve comparable performance, where Mockingjay barely exceeds the performance of MFCC. By adding frequency alteration and magnitude alteration to Mockingjay, TERA can boost the performance by 4.51% over Mockingjay. The best performing representation on keyword spotting is vq-wav2vec, which is surprising as it does not perform as well as the others on phoneme classification and speaker recognition. CPC achieves comparable performance with vq-wav2vec, suggesting that contrastive learning over time may be the key for detection tasks. Other than CPC and vq-wav2vec, VQ-APC and TERA achieve high scores on this task too.

3) Speaker Recognition Results: For the speaker recognition task, TERA achieves the highest speaker classification accuracy for both the *frame* and *utter* setting. We can observe a discriminative comparison under the *frame* setting. Methods that use autoregressive, contrastive, vector-quantization have a lower frame-wise speaker recognition accuracy. The autoregressive prediction method focuses more on the local dependencies between time steps. Also, contrastive learning discriminates input from a distant time, and vector-quantization is a bottleneck that limits information flow over the network. The above techniques seem to have encouraged the model not to encode speaker information in each frame, resulting in poor *frame* classification performance. Although the NPC technique uses time-masked reconstruction like Mockingjay and Audio ALBERT, it performs poorly for the *frame* setting. We speculate that this is because Masked Convolution Blocks' masking is fixed and designed to focus on local dependencies. As a result, each representation doesn't preserve speaker information. On the other hand, the time mask of Mockingjay, Audio ALBERT, and TERA is applied through dynamic masking [20], [21], where masks may overlap each other, creating various time mask lengths. The overlapped masks encourage the model to also focus on global information. As speaker characteristics tend to persist across time, this thus preserves the speaker information. For the *utter* setting of speaker recognition, all of the representations yield satisfactory results except for vq-wav2vec. However, remember that vq-wav2vec achieves the highest score on keyword spotting. We conclude that there is a trade-off for some representation learning methods, vq-wav2vec is good on keyword spotting but lacks speaker information.

C. Analysis on TERA

To better understand how TERA derives representation from speech, we study various aspects of TERA's pre-training. We first investigate how TERA can benefit from fine-tuning, where we present results in Table III. Next, we investigate the effect of increasing the amount of pre-training data. Also, we study the effect of learning with various acoustic features. Furthermore, we investigate how the network depth (number of layers) affects TERA's performance. Finally, we study the difference between directly applying SpecAugment and the TERA time and frequency alteration. We intentionally did not apply the magnitude alteration here to rule out the variation. We visualize results in Figure 4. We denote models trained on 100 hours of LibriSpeech in red and denote models trained on 960 hours of LibriSpeech in blue. All TERA models are pre-trained with time and frequency alteration. We freeze all models for representation extraction from the last layer.

1) Fine-Tuning TERA: In Table III, we show the effect of fine-tuning the pre-trained TERA with downstream tasks. In the last column, we average the accuracy of all tasks across each row. We fine-tune TERA *"time+freq+mag"*, the best performing TERA-base model, which is pre-trained on 960 hours of LibriSpeech. In the first row of Table III, we include the results from the last row of Table II of the frozen TERA *"time+freq+mag"* model to link the two tables. With fine-tuning, we see considerable performance increases for all tasks. By adding SpecAugment [41] during fine-tune, we improve performance for some tasks. However, the improvement of adding SpecAugment during fine-tuning is limited because the pre-trained weights are good initialization for the tasks. Empirically, we find that when

TABLE III
PERFORMANCE OF FINE-TUNING TERA AND BASELINE APPROACHES. HERE WE USE THE TERA-BASE *"TIME+FREQ+MAG"* MODEL, ROW ONE IS IDENTICAL TO THE LAST ROW OF TABLE II. WE REPORT TESTING ACCURACY (%) AND HIGHLIGHT THE HIGHEST SCORE FOR EACH COLUMN IN BOLD FONT

Method	Phoneme - LibriSpeech			Phoneme - TIMIT			Keyword	Speaker		Average
	linear	linear concat	1-hidden	linear	linear concat	1-hidden	spotting	frame	utter	
TERA	74.07	78.67	81.90	74.14	75.92	76.23	92.60	99.47	99.48	83.61
+ *fine-tune*	89.09	89.07	89.53	78.81	**78.86**	78.37	**94.03**	99.50	99.62	88.54
+ *SpecAug*	**90.36**	**90.07**	**90.88**	**79.50**	78.69	**78.82**	93.74	**99.81**	**99.86**	**89.08**
random init + *SpecAug*	88.70	89.49	89.08	70.10	68.09	71.57	8.34	0.424	1.67	54.16

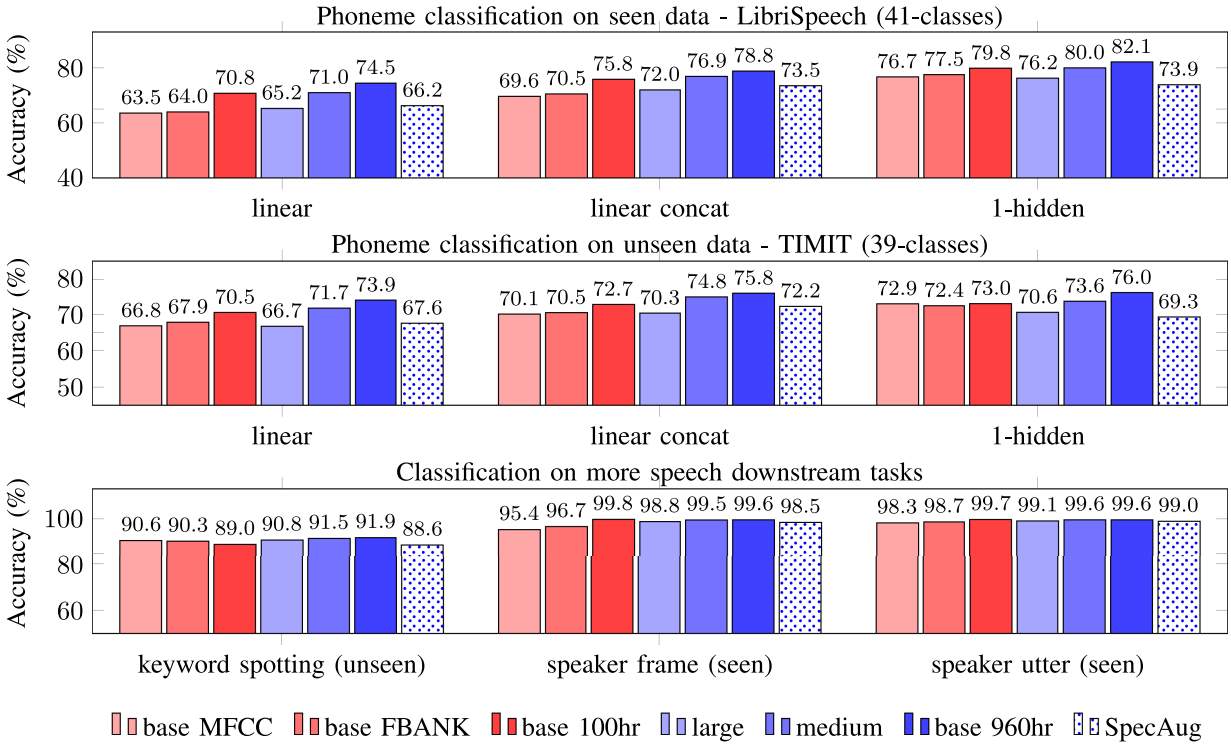

Fig. 4. **Analysis on TERA.** We pre-train TERA with different amount of unlabeled data, with different features, and with different model depth. Moreover, we directly apply the SpecAugment [41] LD Policy for pre-training instead of TERA's alteration, we get performance degrade. The models trained with 100 hours are denoted in red, and the models trained with 960 hours are in denoted blue.

compared to training downstream classifiers on frozen TERA, fine-tuning TERA with downstream tasks dramatically reduces the number of training steps for the downstream models to converge. We also observe that fine-tuning for phoneme classification on LibriSpeech brings a more significant improvement than on TIMIT. The reason is that there are more labeled data for LibriSpeech. It is worth noticing that fine-tuning with limited label data (TIMIT) still benefits performance, where we observe no sign of overfitting.

We train the same TERA network (with randomly initialized parameters) on downstream tasks, where we distort the input features with SpecAugment to prevent overfitting. The above setting serves as a baseline for fine-tuning pre-trained TERA, and we denote it as *random init + SpecAug* in Table III. Overall, the downstream models trained with randomly initialized TERA and SpecAugment either underperform or perform poorly. The baseline *random init + SpecAug* results of phoneme classification on LibriSpeech are close to the pre-trained models because the amount of label is sufficient. However, there is a performance gap between baseline results and pre-trained models on phoneme classification for TIMIT. The reason is the limited amount of label data. The baseline *random init + SpecAug* overfits for the keyword spotting and speaker recognition task. Note that the reported accuracy is from the test set; therefore, a low testing score means severe overfitting on the training set. It does not indicate that the network is untrainable. We conclude that without pre-training, model architecture alone provides no benefit.

2) Pre-Training on More Data: Unsurprisingly, pre-training TERA on more data increases the performance of all downstream tasks. In Figure 4, *base 960hr* improves significantly over *base 100hr*. Presumably, all pre-trained models should benefit from more unlabeled data. However, we find that this is not the case for all methods. In particular, we find that models trained from time-only masked reconstruction (See Table I) may not improve by training on more data, especially when the added data is unclean. To be more specific, we find that Mockingjay [20] and NPC [27] got worse performance when

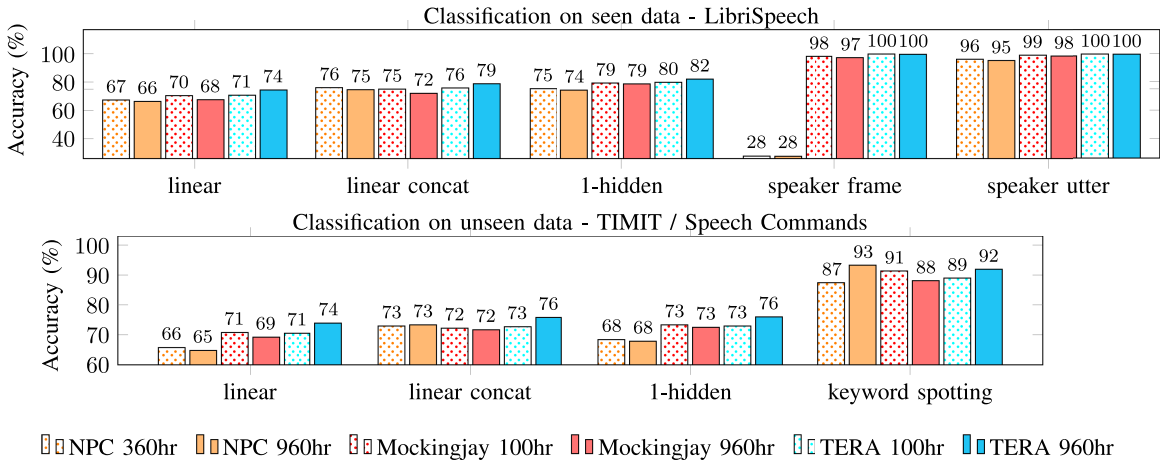

Fig. 5. **Pre-training on more data.** We pre-train different masked reconstruction models on a different amount of unlabeled data. We observe that NPC and Mockingjay got worse performance as we add noisy data (LibriSpeech *train-other-500*) during pre-train, mainly due to the reconstruction nature that remembers everything. TERA, on the other hand, does not suffer from this issue.

we add *train-other-500* to the pre-training data. The Mockingjay method uses dynamic masking on data along the time-axis. The NPC method uses a mask in its convolution module to achieve masking on the time-axis. These methods use the idea of reconstructing a masked time span from surrounding context, which is unlike other pre-training methods such as the line of work of APC [8]–[11] that uses autoregressive prediction and the line of work of CPC [1]–[7] that uses contrastive learning. The time-only masked reconstruction process forces the model to remember all the information of speech, including speaker characteristics and other noise. However, with the regularization of frequency masking, the TERA representations do not exhibit this weakness.

This observation is presented in Figure 5, where NPC [27], Mockingjay [20], and TERA *"time+freq"* are pre-trained with increasing amounts of unlabeled data. We evaluate the learned representations on downstream tasks. For all tasks and all types of classifiers, both NPC and Mockingjay got worse performance when we add the *train-other-500* subset for pre-train. There is only one exception with NPC, where it improves on the keyword spotting task by pre-training on more data. On the other hand, TERA can benefit from training on noisy data. As a result, TERA improves for all tasks by pre-training on more data.

We want to point out that pre-training on more data has not been explored in NPC and Mockingjay, where they only report results with models pre-trained on 360 hours of LibriSpeech. We believe this is a unique phenomenon for methods that learn from the time-only masked reconstruction. To the best of our knowledge, only NPC and Mockingjay examine this kind of behavior. Note that we pre-train Mockingjay and TERA with the same implementation** and setting. The only difference is by adding the frequency masking.

3) Learning With Different Acoustic Features: In Figure 4, we pre-trained TERA with MFCC and FBANK as both input and output targets, instead of log Mel. The setting of these acoustic features is identical to the ones listed in Table I. For phoneme classification on both LibriSpeech and TIMIT, pre-training on log Mel outperforms the other two, while pre-training on FBANK is better than MFCC. By using a more primitive feature, the model can preserve more phoneme information. However, the results are opposite for keyword spotting, where models pre-trained on MFCC slightly outperforms FBANK, and models pre-trained on FBANK slightly exceeds log Mel. Learning from all features successfully preserves the speaker information, with the model pre-trained on log Mel achieving the highest score for speaker recognition. In general, pre-training with log Mel features results in the best performing model. This aligns with previous work [8]–[11], [20], [21], [27] (See Table I) that also uses 80-dim log Mel.

4) Effect of Different Network Depth: We present classification accuracy of three TERA models, including base (3-layer), medium (6-layer), large (12-layer) in Figure 4. We observe that performance decay for all tasks as model depth increases. We conclude that a small model is sufficient to solve the proposed pre-training task, which is reasonable as a lot of work also uses 3-layer models (See Table I). Note that here TERA is used for representation extraction and not fine-tune. In our later ASR experiments, we find that large models outperform small models during fine-tuning. The above discovery aligns with our previous finding with wav2vec 2.0, that large models are excellent for fine-tuning but not representation extraction. On the other hand, small models are adequate for feature extraction than large models.

5) Comparing TERA With SpecAugment: SpecAugment is a regularization technique for improving ASR training [41]. It shares a similar ideology with our time and frequency alteration. We consider the LD Policy of SpecAugment, which is the best performing policy described in the paper. Following the notations in SpecAugment [41], the LD Policy has the following parameters: time mask parameter $T = 100$ (maximum length of the consecutive time mask), frequency mask parameter $F = 9$ (maximum length of the consecutive frequency mask), number of time masks $mT = 2$ (amount of consecutive mask blocks in time), and number of frequency masks $mF = 2$ (amount of consecutive mask blocks in frequency). Here we first point out some key differences between the proposed method and

SpecAugment [41], then we present the experiment results of comparing TERA with SpecAugment.

The difference in time mask length. SpecAugment uses a longer time mask length of up to 100 compared to TERA's length of 7. In SpecAugment, the time mask length applied is chosen from a uniform distribution from 0 to the maximum consecutive length (T) of 100. The considerable variation of time mask length introduces some problems during pre-training. Empirically, we find that pre-training loss of reconstruction from SpecAugment is volatile and not stable. Also, in our experiments, we find that the large variation of time mask length makes it hard for the pre-trained model to encode phonetic information.

The difference in number of time masks. SpecAugment uses two consecutive mask blocks ($mT = 2$) for each input. A fixed amount of consecutive mask blocks is employed. Whereas TERA determines the number of mask blocks by the maximum time alteration percentage $P_T = 15\%$. The amount of mask blocks T_{num} is then given as $T_{num} = \lfloor P_T \times L_x \div W_T \rceil$, where L_x is the length of input. Simply put, the number of masking blocks will change according to different input lengths. The number of time masks will vastly affect the model's training for long utterances. By fixing the number of consecutive mask blocks, SpecAugment does not utilize the advantage of longer training samples.

The difference in masking policy. The SpecAugment masks selected time blocks to zero. In TERA, following the idea of BERT [32], a more sophisticated policy is applied. There are three cases for the selected time blocks, 1) mask to zero, 2) replace with random time blocks, and 3) do nothing. Case 2) helps TERA learn the order of speech and not always reconstruct it from zero, case 3) helps TERA ease the training and testing inconsistency problem. Overall, TERA uses a more advanced time alteration policy, and SpecAugment uses a simple mask-to-zero method.

Comparing experimentally. To compare with SpecAugment experimentally, we apply the SpecAugment LD Policy to self-supervised speech training. We use the same pre-training settings (960 hours) and network architecture for both TERA and SpecAugment (3-layer Transformer Encoders T_{enc} and the 2-layer prediction network P_{net}), with only the difference between masking policies discussed above. We show the results in dark blue and blue dotted color in Figure 4. TERA *base 960hr* largely outperforms *SpecAug* on phoneme classification for both LibriSpeech and TIMIT, and on the keyword spotting task. Also, TERA wins over SpecAugment on the speaker recognition task. We conclude that SpecAugment is suitable for regularizing ASR training but not self-supervised learning. On the other hand, TERA is more effective for self-supervised representation learning.

D. Speech Representations for ASR

We further apply TERA to speech recognition tasks. In Table IV, we list results of TERA and recent work in terms of WER, where all ASR models are trained on top of frozen representations without fine-tuning. All TERA models use a combination of *"time+freq+mag"* alteration as the auxiliary objective. All of the works report WER and LM rescored WER (denoted as Rescore) on the *test-clean* subset of LibriSpeech [45]. All methods are pre-trained with 960 hours of LibriSpeech, except for one experiment setup in [6] where it uses 8000 hours of data for pre-training. All of the work use only the *train-clean-100* for downstream adaption, except for the setup in [6] and [3] that use 96 hours and 960 hours of label, respectively. We use the decoding and rescoring setup described in Section IV-E for all *liGRU + TERA* and its variation, as well as *liGRU* with baseline features. Beam-search decoding with a 4-gram language model is used for Bidir-CPC [6]. For wav2vec-large and DeCoAR [12], a 4-gram LM is used in the first-pass decoding.

TABLE IV
COMPARISON OF RECENT SPEECH REPRESENTATION APPROACHES FOR ASR. ALL RESULTS ARE FROM TRAINING AN ASR SYSTEM ON TOP OF FROZEN REPRESENTATIONS, WITHOUT FINE-TUNING THE PRE-TRAINED MODEL. WE REPORT WER ON THE LIBRISPEECH [45] *TEST-CLEAN* SUBSET

Models	Pre-train	Labels	WER	Rescore
liGRU + MFCC	None	100 hr	8.66	6.42
liGRU + FBANK	None	100 hr	8.64	6.34
liGRU + fMLLR	None	100 hr	8.63	6.25
Bidir-CPC [6]	960 hr	96 hr	14.96	9.41
Bidir-CPC [6]	8000 hr	96 hr	13.69	8.70
vq-wav2vec [3]	960 hr	960 hr	6.2	-
wav2vec-large [12]	960 hr	100 hr	6.92	-
DeCoAR [12]	960 hr	100 hr	6.10	-
liGRU + TERA-base	960 hr	100 hr	**8.31**	**6.01**
liGRU + TERA-medium	960 hr	100 hr	8.37	6.05
liGRU + TERA-large	960 hr	100 hr	8.35	**6.01**

First, we observe that the model sizes of TERA (i.e., *base*, *medium* and *large*) have little influence on the ASR performance when TERA is used as an extractor for speech representation. This observation aligns with our previous discovery on other downstream tasks. The representations from the *base* model are sufficient to improve supervised ASR. Since all the cited models use different LM setups, it is hard to conclude the WER comparison in Table IV. However, we cite other work's performance to show that the WER achieved in this work is well within the expected range. We also investigate three baseline features of MFCC, FBANK, and fMLLR. We use an identical ASR framework and setting of TERA representations for the three features. Our results suggest that TERA yields constant improvement over surface features in the same ASR framework.

E. Speech Pre-Training for ASR Comparison

In this section, we compare the results of fine-tuning various pre-trained models for ASR. All TERA models use a combination of *"time+freq+mag"* alteration as the auxiliary objective. We summarize the results from previous literature as well as fine-tuning TERA with *liGRU* or *MLP* framework in Table V. We also list results from recent literature, where all results are from fine-tuning the pre-trained model as an ASR encoder. Similar to the previous section, we report WER and LM rescored WER on the *test-clean* subset of LibriSpeech [45]. The first-pass decoding and LM rescoring setting are described in Section IV-E. All the methods investigated here were pre-trained with 960 hours and use 100 hours of labels, except for Masked Pre-trained

TABLE V
COMPARISON OF RECENT PRE-TRAINING APPROACHES FOR ASR. ALL RESULTS ARE FROM FINE-TUNING THE PRE-TRAINED MODEL AS SPEECH ENCODERS AS PART OF THE ASR SYSTEM. ASR WER AND WER AFTER LM RESCORING ON THE LIBRISPEECH [45] *TEST-CLEAN* SUBSET ARE REPORTED

Models	Labels	WER	Rescore
wav2vec 2.0 - large [5]	100 hr	**2.3**	-
Discrete BERT + vq-wav2vec [4]	100 hr	**4.5**	-
Continuous BERT + wav2vec [4]	100 hr	11.8	-
Masked Pre-trained Encoders [26]	100 hr	9.68	-
Masked Pre-trained Encoders [26]	*360 hr*	**7.83**	-
liGRU + TERA-base (fine-tune)	100 hr	8.23	5.84
liGRU + TERA-medium (fine-tune)	100 hr	8.22	5.90
liGRU + TERA-large (fine-tune)	100 hr	**8.00**	**5.80**
MLP + TERA-base (fine-tune)	100 hr	8.47	6.24
MLP + TERA-medium (fine-tune)	100 hr	8.02	5.86
MLP + TERA-large (fine-tune)	100 hr	**7.96**	**5.84**

TABLE VI
COMPARISON OF PRE-TRAINING APPROACHES BETWEEN RECENT WORK AND THE PROPOSED APPROACH ON TIMIT [48]. ALL THE PRE-TRAINING DATA ARE FROM LIBRISPEECH [45], IF NOT SPECIFIED OTHERWISE. WE REPORT TESTING RESULTS IN TERMS OF PER. ALL OF THE TERA MODELS USE THE COMBINED AUXILIARY OBJECTIVE OF "TIME+FREQ+MAG" ALTERATION

Models	Pre-train	PER
CNN + TD-filterbanks [56]	None	18.0
CNN + HMM [57]	None	16.5
liGRU + MFCC [58]	None	16.7
liGRU + FBANK [58]	None	15.8
liGRU + fMLLR [58]	None	14.9
wav2vec [2]	80 hr	17.6
wav2vec [2]	960 hr	15.6
wav2vec [2]	960 + WSJ 81 hr	14.7
liGRU + TERA-base	100 hr	15.2
liGRU + TERA-base	360 hr	14.9
liGRU + TERA-base	460 hr	14.9
liGRU + TERA-base	960 hr	**14.5**
liGRU + TERA-base (fine-tune)	960 hr	15.2
MLP + TERA-base (fine-tune)	960 hr	16.6
liGRU + TERA-medium	960 hr	14.9

Encoders [26] trained with 360 hours of labels. The wav2vec 2.0 [5] uses a Transformer [28] language model with beam search size of 500 for decoding. The vq-wav2vec [4] uses a 4-gram LM during first-pass decoding, and Masked Pre-trained Encoders [26] adopt beam search and RNN LM with CTC decoding. When fine-tuning TERA with *liGRU* models, performance roughly correlates with the depth of TERA, and the *large* TERA achieved the best WER. Remember that large models (wav2vec 2.0, TERA-large, etc) did not perform well for feature extraction. However, they are effective for ASR fine-tuning. By comparing the *liGRU* results in Table IV and Table V, we see that fine-tuning TERA consistently outperforms the case when TERA is simply used for extraction of speech representation. The ASR model adopting *base* TERA improves from 6.01% to 5.84%, the *medium* TERA improves from 6.05% to 5.90%, and the *large* TERA from 6.01% to 5.80%.

Here we also cite other work's performance to show that the WER achieved for the proposed method is within the expected range. The wav2vec 2.0 [5] large model achieves a high score of 2.3%. However, we argue that it is consists of seven convolution blocks plus 24 transformer blocks. In contrast, our *base* model contains only a 3-layer Transformer Encoder layers, which brings substantial low-footprint benefits. The massive wav2vec 2.0 model needs to be trained on 128 V100 GPUs, where the TERA model can be trained on a single GPU. The discrete BERT + vq-wav2vec [4] achieves a high score of 4.5%. However, we argue that the two-step pre-training is computation-intensive during model training. First, a discrete vocabulary of the data is learned from vq-wav2vec [3], and then in the second step a standard BERT [32] is trained on these discrete units. Also, the discrete BERT + vq-wav2vec [4] is built by stacking a standard BERT model [32] of 12 Transformer Encoder layers [28] on top of vq-wav2vec [3], which consists of an 8-layer encoder network and a 12-layer aggregator network (or context network, as described in [1]). Our small encoder architecture (3-layer) benefits from less computational cost and can run on edge devices during inference for downstream tasks.

We also fine-tune TERA with *MLP* models, and we find a similar trend but sometimes higher WER compared to TERA with *liGRU*. Using a deeper model with *MLP* gives performance benefit, and *large* achieves the best WER among the *MLP* models. The reason is that the simple architecture of *MLP* can benefit from a deeper TERA model. Comparing *MLP* with *liGRU*, *MLP* achieved superior performance than *liGRU* on the *medium* model, and similar performance for the rest of the model size. Although in general *MLP* outperforms *liGRU*, however *MLP* has the advantage of a fast training and inference time, thanks to the absence of recurrent units. Additionally, the parameters of the 1-layer *MLP* is significantly less than the 5-layer *liGRU* models. To conclude this section for our proposed method, using a deeper model increases ASR performance during fine-tuning.

F. Transferring to TIMIT

We then explore how the mismatch of domains between pre-training and downstream tasks affects performance. For the exploration, we pre-train TERA with LibriSpeech [45], and apply the resulting networks to the supervised TIMIT [48] ASR task. The same Hybrid ASR setting and framework described above for LibriSpeech ASR are used, except that we use a learning rate of $4e^{-4}$ and a batch size of 8. Testing results of TERA and another self-supervised learning technique, wav2vec [2], are summarized in Table VI in terms of PER. We also list the results of strong supervised systems [56]–[58]. All of the TERA models use a combination of *"time+freq+mag"* alteration as the auxiliary objective, and are pre-trained with various amounts of data. As expected, pre-training on a larger amount of data gives performance benefit, and we achieved the best WER (14.5%) with 960 hours of pre-training data. We find that for TIMIT ASR as the downstream task, fine-tuning is not helpful, and extracting speech representations from the last layer provides the best performance. The reason is likely because there is not enough labeled data in TIMIT, which aligns with our discovery in Section V-C1 for other downstream tasks. Also, there is no significant gain when extracting features from a larger model *medium*, which aligns with our previous discussion that smaller models are better for feature extraction.

VI. Conclusion

We propose a novel self-supervised training scheme called TERA, where the model learns from the reconstruction of altered input. We pre-train TERA using a large amount of unlabeled data, and adapt TERA to downstream SLP tasks using a limited amount of labeled data. We demonstrate strong results in phone classification, keyword spotting, speaker recognition, and speech recognition. We conduct a complete ablation study and a thorough comparison of recent representation learning and pre-training approaches. We show that TERA pre-trained on one dataset can be easily transferred to another downstream dataset. We study how self-supervised models behave on more pre-training data and find that time-only masked reconstruction methods cannot benefit from extensive data. We also study the choice of acoustic features for pre-training. We show that it plays a crucial role in reconstruction-based self-supervised learning, as various surface features will lead to significantly different downstream performance. We investigate networks with different depths and find that small models are more suitable for feature extraction than large models. On the other hand, large models are more effective for fine-tuning than small models.

Acknowledgment

The authors are grateful to the National Center for High-performance Computing for computer time and facilities. They thank Shu-wen Yang for implementing a significant part of the S3PRL toolkit and pre-training the APC, VQ-APC, and NPC models; and Yist Y. Lin for implementing the keyword spotting task.

References

[1] A. van den Oord, Y. Li, and O. Vinyals, "Representation Learning With Contrastive Predictive Coding," Comput. Res. Repository, 2018, *arXiv:1807.03748*. [Online]. Available: http://arxiv.org/abs/1807.03748

[2] S. Schneider, A. Baevski, R. Collobert, and M. Auli, "wav2vec: Unsupervised Pre-Training for Speech Recognition," in *Proc. Interspeech*, 2019, pp. 3465–3469.

[3] A. Baevski, S. Schneider, and M. Auli, "vq-wav2vec: Self-supervised learning of discrete speech representations," in *Proc. Int. Conf. Learn. Representations*, 2020. [Online]. Available: https://openreview.net/forum?id=rylwJxrYDS

[4] A. Baevski and A. Mohamed, "Effectiveness of self-supervised pre-training for asr," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process.*, 2020, pp. 7694–7698.

[5] A. Baevski, Y. Zhou, A. Mohamed, and M. Auli, "wav2vec 2.0: A Framework for Self-Supervised Learning of Speech Representations," in NeurIPS, virtual, H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, and H. Lin, Eds., 2020, pp. 12449–12460. [Online]. Available: https://proceedings.neurips.cc/paper/2020/hash/92d1e1eb1cd6f9fba3227870bb6d7f07-Abstract.html

[6] K. Kawakami, L. Wang, C. Dyer, P. Blunsom, and A. van den Oord, "Learning robust and multilingual speech representations," in *Proc. Findings Assoc. Comput. Linguistics: Empirical Methods Natural Lang. Process.* .2020, pp. 1182–1192. [Online]. Available: https://www.aclweb.org/anthology/2020.findings-emnlp.106.

[7] M. Riviére, A. Joulin, P.-E. Mazaré, and E. Dupoux, "Unsupervised pretraining transfers well across languages," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process.*, 2020, pp. 7414–7418.

[8] Y.-A. Chung, W.-N. Hsu, H. Tang, and J. Glass, "An unsupervised autoregressive model for speech representation learning," in *Proc. Interspeech*, 2019, pp. 146–150. [Online]. Available: http://dx.doi.org/10.21437/Interspeech.2019-1473.

[9] Y. Chung and J. Glass, "Generative pre-training for speech with autoregressive predictive coding," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process.*, 2020, pp. 3497–3501.

[10] Y.-A. Chung and J. Glass, "Improved speech representations with multi-target autoregressive predictive coding," in *Proc. 58th Annu. Meeting Assoc. Comput. Linguistics*, 2020, pp. 2353–2358. [Online]. Available: https://www.aclweb.org/anthology/2020.acl-main.213.

[11] Y.-A. Chung, H. Tang, and J. Glass, "Vector-Quantized Autoregressive Predictive Coding," in *Proc. Interspeech*, 2020, pp. 3760–3764.

[12] S. Ling, Y. Liu, J. Salazar, and K. Kirchhoff, "Deep contextualized acoustic representations for semi-supervised speech recognition," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process.*, 2020, pp. 6429–6433.

[13] M. Tagliasacchi, B. Gfeller, F. de Chaumont Quitry, and D. Roblek, "Self-Supervised Audio Representation Learning for Mobile Devices," Comput. Res. Repository , 2019, *arXiv: 1905.11796*. [Online]. Available: http://arxiv.org/abs/1905.11796

[14] M. Tagliasacchi, B. Gfeller, F. d. C. Quitry, and D. Roblek, "Pre-training audio representations with self-supervision," *IEEE Signal Process. Lett.*, vol. 27, pp. 600–604, 2020.

[15] J. Chorowski, R. J. Weiss, S. Bengio, and A. van den Oord, "Unsupervised speech representation learning using wavenet autoencoders," *IEEE/ACM Trans. Audio, Speech, Lang. Process.*, vol. 27, no. 12, pp. 2041–2053, Dec. 2019.

[16] A. T. Liu, P.-c. Hsu, and H.-Y. Lee, "Unsupervised End-to-End Learning of Discrete Linguistic Units for Voice Conversion," in *Proc. Interspeech*, Sep. 2019, pp. 1108–1112.

[17] F. de Chaumont Quitry, M. Tagliasacchi, and D. Roblek, "Learning Audio Representations Via Phase Prediction," 2019.

[18] S. Pascual, M. Ravanelli, J. Serrà, A. Bonafonte, and Y. Bengio, "Learning Problem-Agnostic Speech Representations From Multiple Self-Supervised Tasks," in *Proc. Interspeech*, Sep. 2019, pp. 161–165.

[19] S. Khurana *et al.*, "A Convolutional Deep Markov Model for Unsupervised Speech Representation Learning," in *Proc. Interspeech*, Shanghai, China, Oct. 2020, pp. 3790–3794. [Online]. Available: https://hal.archives-ouvertes.fr/hal-02912029

[20] A. T. Liu, S.-w. Yang, P.-H. Chi, P.-c. Hsu, and H.-y. Lee, "Mockingjay: Unsupervised Speech Representation Learning With Deep Bidirectional Transformer Encoders," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process.*, May 2020, pp. 6419–6423.

[21] P.-H. Chi, P.-H. Chung, T.-H. Wu, C.-C. Hsieh, S.-W. Li, and H. yi Lee, "Audio albert: A lite bert for self-supervised learning of audio representation," in *Proc. IEEE Spoken Lang. Technol. Workshop*, 2020, pp. 344–350.

[22] W. Wang, Q. Tang, and K. Livescu, "Unsupervised Pre-Training of Bidirectional Speech Encoders Via Masked Reconstruction," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process.*, May 2020, pp. 6889–6893.

[23] X. Song, G. Wang, Y. Huang, Z. Wu, D. Su, and H. Meng, "Speech-XLNet: Unsupervised Acoustic Model Pretraining for Self-Attention Networks," in *Proc. Interspeech*, 2020, pp. 3765–3769. [Online]. Available: http://dx.doi.org/10.21437/Interspeech.2020-1511.

[24] S. Li, L. Li, Q. Hong, and L. Liu, "Improving Transformer-Based Speech Recognition With Unsupervised Pre-Training and Multi-Task Semantic Knowledge Learning," in *Proc. Interspeech*, 2020, pp. 5006–5010. [Online]. Available: http://dx.doi.org/10.21437/Interspeech.2020-2007.

[25] D. Jiang *et al.*, "A further study of unsupervised pre-training for transformer based speech recognition," in *Proc. Int. Conf. Learn. Representations, 2021, pp. 6538–6542*. [Online]. Available: https://openreview.net/forum?id=hrpSB_rzQTU

[26] L. Liu and Y. Huang, "Masked Pre-Trained Encoder Base on Joint Ctc-Transformer," 2020.

[27] A. H. Liu, Y.-A. Chung, and J. Glass, "Non-Autoregressive Predictive Coding for Learning Speech Representations From Local Dependencies," 2020.

[28] A. Vaswani *et al.*, "Attention is all you need," in *Proc. 31st Int. Conf. Neural Inf. Process. Syst.*. Red Hook, NY, USA: Curran Associates Inc., 2017, pp. 6000–6010.

[29] C. Sun, X. Qiu, Y. Xu, and X. Huang, "How to fine-tune bert for text classification?" Chinese Comput. Linguistics, vol. 11856, pp. 194–206, 2019.

[30] A. Chronopoulou, C. Baziotis, and A. Potamianos, "An embarrassingly simple approach for transfer learning from pretrained language models," in *Proc. Conf. North*, 2019.

[31] A. T. Liu and Y. Shu-wen, "The S3PRL Toolkit: Self-Supervised Speech Pre-Training and Representation Learning," 2020. [Online]. Available: https://github.com/s3prl/s3prl

[32] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-Training of Deep Bidirectional Transformers for Language Understanding," in *Proc. Conf. North Amer. Chapter Assoc. Comput. Linguistics: Human Lang. Technol.*, vol. 1 (Long and Short Papers). Minneapolis, Minnesota: Assoc. Comput. Linguistics, Jun. 2019, pp. 4171–4186. [Online]. Available: https://www.aclweb.org/anthology/N19-1423.

[33] Y. Liu *et al.*, "A Robustly Optimized BERT Pretraining Approach," CoRR, 2019, *arXiv:1907.11692*. [Online]. Available: https://openreview.net/forum?id=SyxS0T4tvS

[34] A. Graves, S. Fernández, F. Gomez, and J. Schmidhuber, "Connectionist temporal classification: Labelling unsegmented sequence data with recurrent neural networks," in *Proc. 23 rd Int. Conf. Mach. Learn.*, 2006, pp. 369–376.

[35] T. Mikolov, M. Karafiát, L. Burget, J. Černockỳ, and S. Khudanpur, "Recurrent neural network based language model," in *Proc. 11th Annu. Conf. Int. speech Commun. Assoc.*, 2010, pp. 1045–1048.

[36] M. Peters *et al.*, "Deep contextualized word representations," in *Proc. Conf. North Amer. Chapter Assoc. Comput. Linguistics: Human Lang. Technol.*, vol. 1 (*Long Papers)*, 2018, pp. 2227–2237.

[37] Z. Lan, M. Chen, S. Goodman, K. Gimpel, P. Sharma, and R. Soricut, "Albert: A lite bert for self-supervised learning of language representations," in *Proc. Int. Conf. Learn. Representations*, 2020. [Online]. Available: https://openreview.net/forum?id=H1eA7AEtvS

[38] Z. Yang *et al.*, "XLNet: Generalized autoregressive pretraining for language understanding," *Adv. Neural Inf. Process. Syst.*, H. Wallach *et al.*, Eds., Curran Associates, Inc., vol. 32, Jun. 2019.

[39] H. Wu, A. T. Liu, and H. yi Lee, "Defense for black-box attacks on anti-spoofing models by self-supervised learning," in *Proc. Interspeech*, 2020, pp. 3780–3784. [Online]. Available: http://dx.doi.org/10.21437/Interspeech.2020-2026.

[40] S. wen Yang, A. T. Liu, and H. yi Lee, "Understanding self-attention of self-supervised audio transformers," in *Proc. Interspeech*, 2020, pp. 3785–3789. [Online]. Available: http://dx.doi.org/10.21437/Interspeech.2020-2231.

[41] D. S. Park *et al.*, "Specaugment: A Simple Data Augmentation Method for Automatic Speech Recognition," in *Proc. Interspeech*, Sep. 2019, pp. 2613–2617.

[42] P. Wang, L. Wei, Y. Cao, J. Xie, and Z. Nie, "Large-scale unsupervised pre-training for end-to-end spoken language understanding," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process.*, 2020, pp. 7999–8003.

[43] S. Ling, J. Salazar, Y. Liu, and K. Kirchhoff, "BERTphone: Phonetically-aware encoder representations for utterance-level speaker and language recognition," in *Proc. Odyssey Speaker Lang. Recognit. Workshop*, 2020, pp. 9–16.

[44] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," in *Proc. 1st Int. Conf. Learn. Representations*, Scottsdale, Arizona, USA, Workshop Track Proc., Y. Bengio and Y. LeCun, Eds., 2013. [Online]. Available: http://arxiv.org/abs/1301.3781

[45] V. Panayotov, G. Chen, D. Povey, and S. Khudanpur, "Librispeech: An ASR corpus based on public domain audio books," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process.*, 2015, pp. 5206–5210.

[46] N.-Q. Pham, T.-S. Nguyen, J. Niehues, M. Müller, and A. Waibel, "Very Deep Self-Attention Networks for End-to-End Speech Recognition," in Proc. Interspeech, 2019, pp. 66–70. [Online]. Available: http://dx.doi.org/10.21437/Interspeech.2019-2702.

[47] I. Loshchilov and F. Hutter, "Decoupled weight decay regularization," in Proc. *Int. Conf. Learn. Representations*, 2019. [Online]. Available: https://openreview.net/forum?id=Bkg6RiCqY7

[48] J. S. Garofolo, L. F. Lamel, W. M. Fisher, J. G. Fiscus, and D. S. Pallett, "DARPA TIMIT acoustic-phonetic continous speech corpus CD-ROM. NIST speech disc 1-1.1," NASA STI/Recon Tech. Rep., Feb. 1993, Art no. 27403.

[49] D. Povey *et al.*, "The Kaldi Speech Recognition Toolkit," in Proc. Autom. Speech Recognit. Understanding, 2011.

[50] C. Lopes and F. Perdigao, "Phone recognition on the timit database," *Speech Technol./Book*, vol. 1, 2011, pp. 285–302.

[51] P. Warden, "Speech Commands: A Public Dataset for Single-Word Speech Recognition." Dataset available online, 2017. [Online]. Available: http://download.tensorflow.org/data/speech_commands_v0.01.tar.gz.

[52] K.-F. Lee and H.-W. Hon, "Speaker-independent phone recognition using hidden markov models," *IEEE Trans. Acoust., Speech, Signal Process.*, vol. 37, no. 11, pp. 1641–1648, Nov. 1989.

[53] M. Ravanelli, T. Parcollet, and Y. Bengio, "The pytorch-kaldi speech recognition toolkit," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process.*, 2019, pp. 6465–6469.

[54] M. J. Gales, "Maximum likelihood linear transformations for hmm-based speech recognition," *Comput. Speech Lang.*, vol. 12, no. 2, pp. 75–98, 1998.

[55] J. Kahn *et al.*, "Libri-Light: A Benchmark for Asr With Limited or No Supervision," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process.*, 2020, pp. 7669–7673, https://github.com/facebookresearch/libri-light.

[56] N. Zeghidour, N. Usunier, I. Kokkinos, T. Schaiz, G. Synnaeve, and E. Dupoux, "Learning Filterbanks From Raw Speech for Phone Recognition," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process.*, Apr.2018, pp. 5509–5513.

[57] L. Tóth, "Phone recognition with hierarchical convolutional deep maxout networks," *EURASIP J. Audio, Speech, Music Process.*, vol. 2015, no. 1, pp. 1–13, 2015.

[58] M. Ravanelli, P. Brakel, M. Omologo, and Y. Bengio, "Light gated recurrent units for speech recognition," *IEEE Trans. Emerg. Topics Comput. Intell.*, vol. 2, no. 2, pp. 92–102, Apr. 2018.

Andy T. Liu received the bachelor's degree in electrical engineering from National Taiwan University (NTU), Taipei, Taiwan, in 2018. He is currently working toward the Ph.D. degree with the Graduate Institute of Communication Engineering, NTU, supervised by Professor Hung-yi Lee. His research interests include self-supervised learning, pre-training, and representation learning in the speech and NLP domain.

Shang-Wen Li received the Ph.D. degree from MIT Computer Science and Artificial Intelligence Laboratory, Cambridge, MA, USA, in 2016 supervised by Professor Victor Zue. Since 2019, he has been a Senior Applied Scientist with Amazon AWS AI. He was with Apple Siri and Amazon Alexa before joining AWS. His research interests include spoken language understanding, natural language generation, dialog management, and low-resource speech processing.

Hung-yi Lee received the M.S. and Ph.D. degrees from National Taiwan University, Taipei, Taiwan, in 2010 and 2012, respectively. From September 2012 to August 2013, he was a Postdoctoral Fellow with the Research Center for Information Technology Innovation, Academia Sinica, Taipei, Taiwan. From September 2013 to July 2014, he was a Visiting Scientist with Spoken Language Systems Group, MIT Computer Science and Artificial Intelligence Laboratory, Cambridge, MA, USA. He is currently an Assistant Professor with the Department of Electrical Engineering, National Taiwan University, with a joint appointment with the Department of Computer Science and Information Engineering of the university. His research interests include spoken language understanding, speech recognition, and machine learning.