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 through the formulation of a single auxiliary task like contrastive prediction, autoregressive prediction, or masked reconstruction. Unlike previous approaches, we use a multi-target auxiliary task to pre-train Transformer Encoders on a large amount of unlabeled speech. The model learns through the reconstruction of acoustic frames from its altered counterpart, where we use a stochastic policy to alter along three dimensions: temporal, channel, and magnitude. TERA can be used to extract speech representations or fine-tune with downstream models. We evaluate TERA on several downstream tasks, including phoneme classification, speaker recognition, and speech recognition. TERA achieved strong performance on these tasks by improving upon surface features and outperforming previous methods. In our experiments, we show that through alteration along different dimensions, the model learns to encode distinct aspects of speech. We explore different knowledge transfer methods to incorporate the pre-trained model with downstream models. Furthermore, we show that the proposed method can be easily transferred to another dataset not used in pre-training.**

***Index Terms*—self-supervised, pre-training, representation**

I. Introduction

Unlike humans, who are 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. In order for the next generation of speech processing systems to exhibit similar levels of cognitive intelligence 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 a large amount of unlabeled data. Self-supervised learning leverage unsupervised pre-training tasks to train networks, and they have shown to be effective for improving downstream systems [1]–[26].

Through 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]–[4], [6]–[21]. 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 good initialization for ASR encoders [5], [20]–[26].

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 that can be potentially transferred to multiple downstream tasks. The key 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 hard enough for the model to learn high-level semantic properties, and not be too amiable for the model to exploit low-level shortcuts.

In this work, we propose TERA: Transformer Encoder Representations from Alteration, a multi-target auxiliary objective to pre-train Transformer Encoders [27]. We introduce a total of three auxiliary objectives to form the multi-target pre-training scheme: 1) time alteration: reconstructing from corrupted blocks of time steps. 2) channel alteration: reconstructing from missing blocks of frequency channels. 3) magnitude alteration: reconstructing from altered feature magnitudes. These auxiliary objectives can be applied together or separately in the pre-training process. 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. Similar self-supervised frameworks have been widely studied (Section II provides a thorough review). Unlike previous approaches that only employ reconstruction on the temporal axis, TERA considers three orthogonal axes, including temporal, channel, and magnitude.

To evaluate TERA, we use downstream tasks of phoneme classification, speaker recognition, and automatic speech recognition (ASR). Also, we compare the effectiveness of each auxiliary objectives separately and in combination. As a result, we confirm that each of the proposed auxiliary objectives guides the model to learn a distinct aspect of speech: 1) The time alteration objective is effective in making more accurate phoneme prediction and speech recognition, as it leads the model to learn richer phonetic content. 2) The channel alter-

A. Liu and H. Lee are with the Graduate Institute of Communication Engineering, College of Electrical Engineering and Computer Science, National Taiwan University. Email: {r07942089, hungyilee}@ntu.edu.tw

S. Li is with Amazon AI. Work done before joining Amazon. Email: shangwel@amazon.com

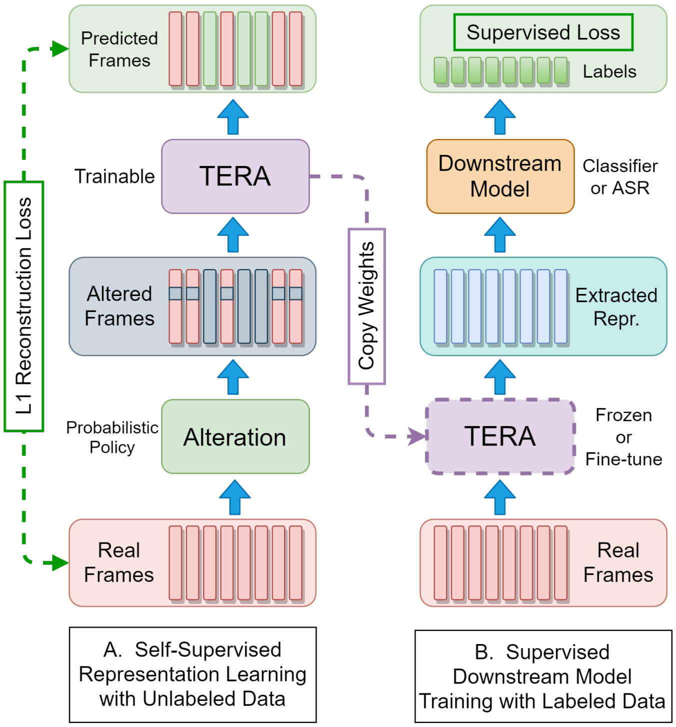

Fig. 1: The illustration of the proposed TERA self-supervised speech representation approach.

ation objective is effective in making more accurate speaker prediction, as it leads the model to learn speaker identity. 3) The magnitude alteration objective is effective in providing a performance boost for all tasks, as it potentially increases data diversity for pre-training.

Besides, we explore different knowledge transfer methods of the pre-trained model to downstream tasks. The methods include: 1) extract representations from the last layer, 2) combine representations from all hidden layers with a learnable weighted sum, and 3) fine-tuning the pre-trained model with the downstream model. Furthermore, we also explore using different acoustic features for reconstruction and find that they impact downstream performance and affect what the model learns. Finally, we investigate the problem of domain mismatch between the pre-training and downstream datasets, and the proposed approach is shown to be unaffected by the domain mismatch issue. For reproducibility of our results, we provide our implementation with pre-trained models and evaluation scripts in the S3PRL [28] toolkit[1].

II. Related Work

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

A. Contrastive Losses

1) CPC: The CPC paper [1], [7] 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 other words, the contrastive loss pulls temporally nearby representations closer and pushes temporally distant ones further. In wav2vec [3], the CPC [1] loss is used to pre-train speech representations for the purpose of speech recognition, and experiment results show self-supervised pre-training improves supervised speech recognition. Also, the CPC loss can be used to regularize adversarial training [2]. The CPC loss has also been extended and applied to bidirectional context networks [6].

2) CPC with Quantization: In vq-wav2vec [4], the wav2vec [3] approach is incorporated with the well-performing Natural Language Processing (NLP) algorithm – Bidirectional Encoder Representations from Transformers (BERT) [29], [30]. The vq-wav2vec [4] approach learns BERT speech representations through a two-stage training pipeline. In the first stage, input speech is discretized to a K-way quantized embedding space by utilizing the wav2vec [3] loss and architecture. Through vector quantization (VQ), continuous speech could act like discrete units similar to word tokens in NLP tasks. These VQ tokens thus enable the direct application of NLP algorithms, which require discrete input. In the second stage, a standard BERT model is trained on top of the VQ discretized tokens to extract speech representations. In a follow-up work [5], the pre-trained vq-wav2vec [4] model is directly fine-tuned on transcribed speech using a Connectionist Temporal Classification (CTC) [31] loss instead of feeding the representations into a task-specific model.

In this work, TERA is compared with CPC [1] and Modified CPC [7] by reporting benchmarking results on LibriSpeech phone classification. We also compare with Bidir-CPC [6], wav2vec [3], vq-wav2vec [4], and BERT + vq-wav2vec [5] by evaluating ASR in terms of phone error rate (PER) and word error rate (WER).

B. Reconstruction Losses

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

1) APC: Largely 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 [32] while conditioning on past frames. Thus, the APC auxiliary task's objective is to reconstruct the future frames conditioning on the 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 memory through reconstruction. 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. Combining the bidirectionality of ELMo [33] and the reconstruction objective of APC [8], [9], in recent literature models were able to learn deep contextualized acoustic representations, DeCoAR [12]. In this work, we also compare TERA with the bidirectional DeCoAR [12] approach.

[1]The S3PRL Toolkit: https://github.com/andi611/Self-Supervised-Speech-Pretraining-and-Representation-Learning

2) BERT-style Masked Reconstruction: Inspired by the Masked Language Model (MLM) task from BERT [29], [30], [34] and Permutation Language Modeling (PLM) from XLNet [35], recent work [20]–[26] have explored using BERT-style tasks to pre-train speech encoders. These approaches adapt the NLP pre-training technique to continuous speech. In Mockingjay [20], input frames of speech are masked to zero to pre-train Transformer Encoders. The masking policy is similar to BERT [29] and RoBERTa [30]. In Audio ALBERT [21], Mockingjay is modified to have shared parameters across Transformer layers. In [36], Mockingjay is shown to be effective in defending adversarial black-box attacks. And in [37], the self-attention of Mockingjay is shown to be meaningful and explainable. Whereas TERA can be seen as an extended version of Mockingjay [20]. Using the time alteration objective along reduces TERA to Mockingjay.

In [24], [26], BERT-style 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 [35], models learn by reconstructing from shuffled input speech frame orders rather than masked frames. In [22], SpecAugment [38] is applied on input frames to pre-train ASR encoders (bi-GRUs). In [39], phoneme posterior vectors are used to train a standard BERT [29], [35] model. The phoneme posterior vectors are output from a supervised acoustic model, which requires CTC loss training over the ground-truth phonemes. Also, in [40], CTC loss is used along with BERT-style mask reconstruction training to learn phonetic representations. As [39] and [40] 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 APC-style and BERT-style losses, previous works have also explored 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 [41] 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. In these works, the autoencoder framework is designed to encode only phonetic content in latent representation, and remove other confounding detail such as speaker identity. The learned representations are then extracted as latent code from the encoder output. 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.

C. Contribution of this Work

The design of the auxiliary task fundamentally decides what the model learns through its reconstruction. Previous work explored mostly for reconstruction on the temporal axis, for example unidirectional 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]–[26]. This work contrasts with prior work in several ways. Firstly, unlike previous work that only employs reconstruction on the temporal axis, we formulate auxiliary objectives with reconstruction loss along three orthogonal axes, including temporal, channel, and magnitude axis. Secondly, most works evaluated their approach with classification tasks [1], [7], [8], [11], [13]–[17], [20], [21], in contrast, we moved beyond classification and applied our model to ASR. Thirdly, we explore knowledge transfer between pre-trained models and downstream tasks, which is an under-investigated problem in speech compared to NLP [42]–[44]. For a comprehensive investigation, we leverage three ways to incorporate the pre-trained model with downstream tasks. Most of the previous work only explored one way of transferring their pre-trained models. Additionally, we propose the use of fMLLR, which is not explored before, as the reconstruction input and target. We also explore pre-training on other types of acoustic features, including MFCC and FBANK. We would like to point out that none of the previous work explores more than one acoustic feature for their method. In our study, we find that the use of different acoustic features in reconstruction-based learning has a large effect on pre-trained models, and is a parameter choice that researchers have to decide. Finally, we show explicitly that our approach continues to work well in the face of domain mismatch between pre-training and downstream datasets.

III. Proposed Methodology

A. Multi-target Auxiliary Objective

As illustrated in Fig. 1A, 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, or fMLLR. We show a sample of 40-dimensional fMLLR feature sequence from the LibriSpeech [45] *train-clean-100* subset in Fig. 2A. 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 channels) of $\overrightarrow{x}$ is denoted as L_x and H_x, respectively. Below, we introduce how we use different auxiliary objectives to alter these $\overrightarrow{x}$.

1) Time Alteration: Through the alteration of contiguous segments along the time axis, our model learns bidirectional representations from past and future context. In this auxiliary objective, a certain percentage of input frames are altered during training, and the model attempts to reconstruct the corrupted span from neighboring frames. To alter the input utterance, we randomly select T_{num} amount of starting locations I_T without replacement. The amount T_{num} is given as

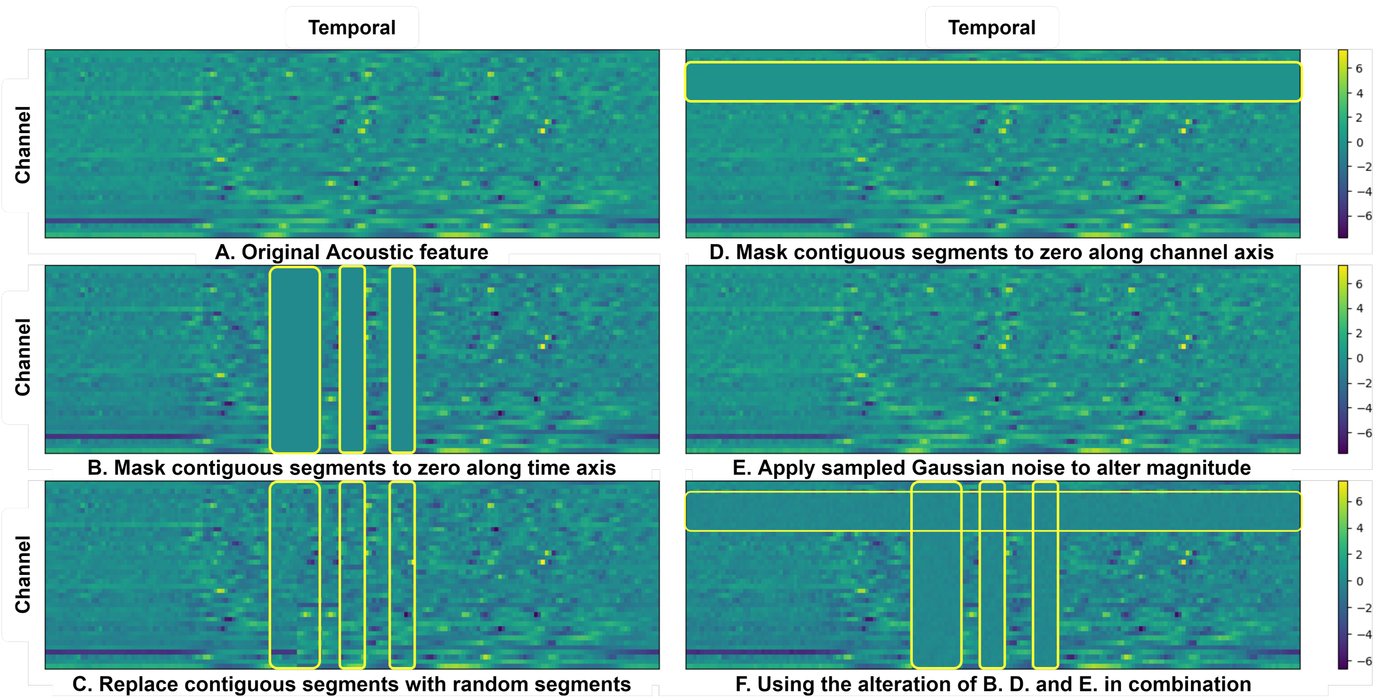

Fig. 2: The illustration of different inputs with various alteration applied for the proposed auxiliary objective. The altered part is highlighted in yellow.

the maximum time alteration percentage P_T normalized by the time alteration width W_T:

$$T_{num} = \lfloor P_T \times L_x \div W_T \rceil \quad (1)$$

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 results from that the model will only receive acoustic features without alteration during inference time.

We illustrate the masking and replacing of frames in Fig. 2B and 2C, respectively. Our time alteration policy is more sophisticated than other BERT-style mask 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 85ms of speech, this lies in the range of an average phoneme duration (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], [29], [30]. We allow time alteration blocks to overlap each other, hence resulting in the larger highlighted yellow box in the left of Fig. 2B and 2C. 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 segments of speech should provide a contextualized understanding of previous and future content. Reconstructing corrupted blocks in the temporal axis is reminiscent of the MLM task in the BERT-style pre-training [29], [30], [34] from NLP, where the models are required to predict the "MASK" token from neighboring tokens. In our ablation study in Section V-F2, we show that the proposed time alteration is the key element that drives models to learn bidirectional understanding, resulting in a substantial WER drop when compared to models that did not use the time alteration objective.

2) Channel Alteration: Our second auxiliary objective is channel alteration. It is largely inspired by SpecAugment proposed for speech augmentation [38], and the ASR pre-training scheme proposed in [22]. For this objective, we randomly mask the values of a block of consecutive channels to zero for all time steps across the input sequence. The block of masked channels is selected by first sampling the width of block W_C from $\{0, 1, ..., W_C\}$ uniformly. Then, we sample a channel index I_C from $\{0, 1, ..., H_x - W_c - 1\}$, where H_x is the number of channels in input sequence $\overrightarrow{x}$. The channels from I_C to $I_C + W_c - 1$ are those to be masked. Note that for $1/(W_C + 1)$ of the time, none of the channels will be masked. Thus, from time to time the model will receive inputs with all of the channel information. This addressed the inconsistency between training and inference time.

We illustrate the effect of this objective on input sequence in Fig. 2D. Unlike the time alteration case, where we sample a number of blocks for alteration as visualized in Fig. 2B and 2C, we only sample a single block for channel alternation 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 channels H_x. Hence, we select multiple blocks for alteration along the time axis, but only one for the channel axis. Following the work of [22], we set the maximum channel alteration width W_C to 8 channels. The intuition behind channel alteration is that a model that can predict the partial loss of channel information should learn a high-level understanding along the channel axis.

As we will show in Section V-F3, through ablation study we find that the proposed channel alteration is the largest contributor in learning speaker identity. Using channel objective provides a more linearly spreadable speaker representation, and a stronger speaker recognizer. Surprisingly, encoding speaker information through this objective does not compromise ASR performance. Which is counter-intuitive as many previous studies [16], [47] pointed out that the key to a successful ASR is to remove speaker variability and preserves only the content of speech. 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 objective, magnitude alteration, by applying sampled Gaussian noise to augment the magnitude of input sequences 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 $\overrightarrow{x}$. This is shown in Fig. 2E, where magnitude alteration was applied 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 [38]). Additionally, magnitude alteration offers another variation to all the 'mask to zero' cases described in Section III-A1 (Time Alteration) and Section III-A2 (Channel Alteration). We illustrate this new 'mask to noise' variation in Fig. 2F, where the selected blocks of time and channel are now with random magnitudes instead of zeros. Empirically, altering magnitude provides a performance benefit for all downstream tasks, as it increases the variation of input data. Also, the gain from magnitude alteration is additive to that of other alteration objectives.

B. Pre-training TERA

We use three auxiliary objectives to build the TERA multi-target pre-training task, where the model is required to minimize the reconstruction error of acoustic features given altered frames as input. The proposed three auxiliary objectives can be used separately or used together as a mixture, as shown in Fig. 2. A stochastic alteration policy samples random pattern based on the applied alterations every time we feed an input sequence to the model. We denote the altered input as $\hat{x}$. 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. 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 four model sizes: *base* (Layer=3, Parameters=21.3M), *medium* (Layer=6, Parameters=42.6M), *large* (Layer=12, Parameters=85.1M), and *xlarge* (Layer=24, Parameters=170.1M). *Medium* was chosen to have the same amount of layer as CPC [1] and Modified CPC [7] for comparison. The Transformer Encoders T_{enc} and the prediction network P_{net} are concatenated to reconstruct $\overrightarrow{x}$ from $\hat{x}$. Our implementation and pre-trained models are available online[1].

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 12 to find model parameters that minimize the L1 loss under the multi-target pre-training task. The Adam optimizer [48] is employed for updating model parameters, where 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 11GB of memory. This allows interested parties to easily train our model with their own data, without the need of massive computational resource. 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.

C. Incorporating with Downstream Tasks

There are many ways to incorporate the learned TERA model to downstream tasks.

1) Representation Extraction: The first approach is to extract representations from the deepest layer of TERA, 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 is 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 *FT* (fine-tune) in later experiments.

3) Weighted Sum: The third approach is to leverage information from all layers since different layers of neural network tend to capture different information [8], [25], and the transferability of different layers also varies a lot [49], [50]. We expose a mixture of representations from all layers of TERA to downstream models, where the mixture is obtained through a learnable weighted sum. This way, how the information encoded in each layer is used can be adapted according to the target downstream task, as each layer's degree of participation is learned from data. This method of combining representation from various layers is also used in ELMo [33] and Mockingjay [20]. We denote this approach as *WS* (weighted sum) in later experiments. This approach can be combined with either representation extraction or fine-tuning.

For both representation extraction and fine-tuning (with or without *WS*), the Adam [48] optimizer is used to update models when training with downstream classification tasks. We use a learning rate of $4e^{-3}$ with a batch size of 6. 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 development set error does not drop more than a threshold of 0.001. A batch size of 16 is used, and we update for 24 epochs. As for applying fine-tuning to ASR, we use the same setting as above, except that a different learning rate for each TERA model is utilized. 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 a larger model requires a lower learning rate during fine-tuning. Hence, we half the learning rate every time the model depth is doubled. On a large degree, this stabilizes the entire fine-tuning process. We did not fine-tune *xlarge* since 24 layers are too unstable to fine-tune.

During ASR fine-tuning, we also apply a modified SpecAugment [38] policy. SpecAugment is a regularization technique for fine-tuning pre-trained models with ASR [22]. Following the notations in SpecAugment [38], we use a time mask parameter of *T=70* (length of the consecutive time mask), frequency mask parameter of *F=4* (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). The chosen (and possibly overlapping) spans of time frames and channels are then zeroed out. We omit the use of time warping since masking. We find SpecAugment is addictive to the proposed pre-training approach, as it delays overfitting and improves the final accuracy numbers in the downstream ASR task. Please refer to the original paper of SpecAugment [38] for a detailed explanation. For reproducibility, we provide ASR training scripts[1] in the format of standard PyTorch-Kaldi [51] configuration files.

IV. Experimental Setup

We use three downstream tasks for evaluation: phone classification, speaker recognition, and speech recognition.

A. Datasets

For most of our experiments, we use the publicly available LibriSpeech [45] corpus. We consider three subsets of LibriSpeech for pre-training, the *train-clean-100*, the *train-clean-360*, and the *train-other-500* subset. We use these subsets to form speech data collections of various sizes, including 100 hours (*train-clean-100*), 360 hours (*train-clean-360*), 460 hours (*train-clean-100* + *train-clean-360*), and the entire 960 hours (*train-clean-100* + *train-clean-360* + *train-other-500*) of LibriSpeech [45]. We set the total training steps T_{steps} of TERA as 200k, 500k, 500k, 1M for 100 hours, 360 hours, 460 hours, 960 hours of data, respectively. We also use TIMIT [52] to evaluate the transferability of pre-trained models. We consider three sets of TIMIT for ASR, the *training* set, the *development* set, and the *testing* set.

In this work, the input to our models is 40-dimensional fMLLR features if not specified otherwise. We also explore pre-training with other acoustic features, including 39-dimensional MFCC and 80-dimensional FBANK. All of the features are extracted as reported in the s5 recipe of Kaldi [53], using windows of 25 ms and an overlap of 10 ms. We apply per-speaker CMVN (cepstral mean and variance normalization) to the features.

B. Phoneme classification Setup

We measure the phoneme prediction performance with classifiers trained on top of TERA representations. Following previous work [1], [7], we adopt the common setup using 41 possible phoneme classes and the *train-clean-100* subset of LibriSpeech [45]. For a fair comparison, we use aligned phoneme labels and train/test split provided in the CPC [1] and Modified CPC [7]. Following the previous work [1], we utilize linear classifiers to measure the linear separability of phonemes. These classifiers are denoted as *Linear*. Additionally, in order to compare with 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 *Concat*. As not all the information encoded is linearly accessible, in addition to measuring linear separability, we also evaluated classifiers with a single hidden layer, following the same settings in CPC [1]. We denote such setting as *1 Hidden*. We intentionally use publicly known settings described above to link our works with previous ones, and to allow easier comparison.

C. Speaker classification Setup

We evaluate TERA representations on speaker prediction tasks. Following the common experiment setting [1], [20], [21], we also adopt the LibriSpeech [45] *train-clean-100* subset, which consists of 251 speakers. We use the same train/test split as provided in the CPC literature [1]. The pre-trained models are evaluated with two types of speaker classification tasks, frame-wise linear classification and utterance-wise linear classification. For frame-wise speaker classification, the classifier predicts speaker for each input frame. This method is denoted as *Frame*. As for utterance-wise classification, representation of each utterance is first averaged over time, then the classifier predicts speaker identity conditioning on the averaged vector. We denote this experiment setting as *Utterance*. In general, the *Frame* classification task is more difficult than *Utterance*, while *Utterance* is a more common scenario for speaker classification. Here we report both *Frame* and *Utterance* for completeness. For both tasks, we employ a linear classification model. These two speaker classification tasks are also investigated in [1], [21]. We note that speaker classification on LibriSpeech only serves as a sanity check for the presence of speaker identity [1], [8], [11], [20], [21].

D. Hybrid DNN/HMM ASR Setup

We evaluate performance of ASR models built on top of TERA representations. We employ the Hybrid DNN/HMM

Representation \ Pre-train	100 hr		360 hr		960 hr	
	Linear	1 Hidden	Linear	1 Hidden	Linear	1 Hidden
CPC [1]	64.6	72.5	-			
TERA-base: time (Mockingjay [20])	64.3	76.8	64.4	77.0	**67.0**	**79.1**
TERA-base: time + mag	64.1	77.1	64.5	77.3	64.7	77.8
TERA-base: time + channel	**65.2**	**77.4**	66.0	78.1	65.9	78.5
TERA-base: time + channel + mag	65.1	77.3	**66.4**	**78.3**	66.4	78.9
MFCC	39.7	59.9	← surface features with *Linear* classifier			
FBANK	42.1	46.9				
fMLLR	52.6	68.4				

TABLE I: **Frame-wise phone classification results on LibriSpeech.** The training and testing sets are all identical to the ones used in the CPC [1] literature. We present testing set accuracy (%) from *Linear* classifiers and *1 Hidden* layer classifiers, measured across various auxiliary objectives and amount of pre-training data.

Representation	#L	#Param \ Pre-train	100 hr		360 hr		960 hr	
			Linear	1 Hidden	Linear	Concat	Linear	1 Hidden
CPC [1] [7]	6	-	64.6	72.5	-	65.5	-	
Modified CPC [7]	6	-	-		-	68.9	-	
TERA-base	3	21.3M	65.1	77.3	66.4	68.3	66.4	**78.9**
TERA-medium	6	42.6M	65.9	77.5	66.6	68.9	**67.3**	78.8
TERA-large	12	85.1M	66.8	**77.7**	**67.5**	**71.7**	67.2	78.5
TERA-xlarge	24	170.1M	**66.9**	77.6	67.1	71.2	**67.3**	78.3

TABLE II: **Comparison of different network depth and size.** LibriSpeech [45] frame-wise phone classification results are presented, with the same phone set, train and test set as in Table I.

ASR modeling implemented with the PyTorch-Kaldi [51] 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 units followed by 2-layers of fully-connected network. 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, the TERA model is updated together with *liGRU* or *MLP* as part of the DNN component in the hybrid ASR framework. We report ASR modelling results of LibriSpeech in terms of WER. To highlight the effect of pre-training, we use 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. In addition to evaluating on LibriSpeech, we also benchmark ASR results with TIMIT, where performance is measured in terms of PER. With this setup, we investigate the issue of domain shift, where models are pre-trained using data in the domains different from the downstream tasks (LibriSpeech and TIMIT in our case). Following the conventional settings [54], ASR modelling on TIMIT is based on 48 phoneme classes, while accuracy is measured after mapping the prediction to a smaller set of 39 phoneme classes. All the results we report here were obtained by training on the training set, tuning hyperparameter on the development set, and testing on the test set. Splits of these sets are obtained from the Kaldi TIMIT recipe [53].

V. Results

In Section V-A and V-B, we study the capability of our pre-trained model in encoding phonetic content and speaker identity, respetively. Then in Section V-C, we evaluate extracted speech representations with ASR models on LibriSpeech [45]. In Section V-D, we compare TERA with approaches where ASR models are trained with frozen speech representations. In Section V-E, TERA is compared with approaches that fine-tune their pre-trained models. Furthermore, in Section V-F, we present the results of our ablation study. In Section V-G, we demonstrate transferring TERA representations from one dataset to another by presenting ASR PER on TIMIT [52].

A. Evaluating Learned Phonetic Content

We present frame-wise phone classification results in Table I in terms of accuracy. We show results of TERA pre-trained with different combinations of objectives, and with different amount of pre-training data. TERA here is used for representation extraction (from the last layer), i.e., TERA parameters were frozen when adapting classifiers to phoneme classification. The classification accuracy is compared with four different baselines: MFCC, FBANK, fMLLR features, and CPC representations [1]. For the *Linear* classifier, FBANK outperforms MFCC, and fMLLR outperforms FBANK in terms of linear separability. For the *1 Hidden* classifier, fMLLR again achieves the highest performance among surface features. On the other hand, as expected TERA and CPC representations outperform all surface features for both types of classifiers.

In the case where models are pre-trained with 100 hours of data, TERA representation (65.2% / 77.4%) outperforms CPC (64.6% / 72.5%) when two or more objectives are applied. We observe that as more objectives are utilized, separability of learned TERA representation increases. When only using *time* as the objective, TERA is equivalent to Mockingjay [20]. By adding the *channel* objective to pre-training, performance is increased. Adding the *magnitude* objective helps performance in some scenarios, and yields comparable results in the others.

Representation \ Pre-train	100 hr		360 hr		960 hr	
	Frame	Utterance	Frame	Utterance	Frame	Utterance
CPC [1]	97.4	-	-			
AALBERT-3L [21]	-		98.8	99.1	-	
TERA-base: time (Mockingjay [20])	68.4	96.1	86.9	97.3	99.3	99.7
TERA-base: time + mag	70.8	96.1	88.0	98.0	99.2	98.8
TERA-base: time + channel	93.6	98.5	**99.4**	**99.5**	**99.5**	**99.8**
TERA-base: time + channel + mag	**98.9**	**99.2**	99.0	**99.5**	99.4	99.8
MFCC	17.6	10.8	← surface features with *Linear* classifier			
FBANK	0.6	5.4				
fMLLR	0.4	2.6				

TABLE III: **Speaker linear classification results on LibriSpeech.** We use identical training and testing sets as in the CPC [1] and AALBERT [21] literature. We show both frame-wise linear classification accuracy (*Frame*, %), and utterance-wise linear classification accuracy (*Utterance*, %) where we average representations over time.

Models \ Pre-train	100 hr		460 hr		960 hr	
	WER	Rescore	WER	Rescore	WER	Rescore
liGRU + TERA-base: time (Mockingjay [20])	8.46	6.12	8.38	6.10	**8.31**	**5.99**
liGRU + TERA-base: time + mag	8.43	6.11	8.38	6.04	8.40	6.03
liGRU + TERA-base: time + channel	8.57	6.16	8.49	6.08	8.35	6.07
liGRU + TERA-base: time + channel + mag	**8.32**	**6.01**	**8.29**	**6.00**	**8.31**	6.01

TABLE IV: **We investigate how different amount of pre-training data affects performance by reporting ASR results on the LibriSpeech [45] *test-clean* subset.** For all the models, we limit the amount of labeled data to 100 hours to demonstrate the effect of pre-training.

As we increase the amount of pre-training data, in general, TERA yields better performance. The trend is consistent for all different combinations of TERA objectives. Comparing the results from pre-training TERA with 100/360 hours of data with the one using 960 hours, we found that when the data is not large enough, utilizing more training objectives tends to improve phonetic separability. The reason is that having more alteration in input augments the pre-training data. The results show that increasing data diversity through TERA alteration objectives can effectively compensate scenarios where small amount of pre-training data is available. For the case where TERA is pre-trained with 960 hours of data, since the corpus is large and diverge enough, only using the *time* objective is sufficient to obtain the best performance (67.0% / 79.1%), and using all of the objectives simply gives comparable results (66.4% / 78.9%). Comparing the *Linear* and *1 Hidden* classifiers, *1 Hidden* classifier outperforms *Linear* in all of the experiment configuration. This result is as expected, as not all information is linearly accessible.

Furthermore, we investigate how the network depth (number of layers) affects the performance of TERA in phoneme classification. In Table II, we summarize phoneme classification results for various network depth. Classification accuracy of four TERA models: *base*, *medium*, *large*, and *xlarge*, is presented together with CPC [1] and Modified CPC [7]. The scores of CPC and Modified CPC are from the original literature. All of the TERA models use the full objective of *time* + *channel* + *mag*. For the case where models are pre-trained with 100 hours of data, all the TERA models outperformed CPC. When 360 hours of data is used for pre-training, we report *Concat* instead of *1 Hidden* to compare TERA with Modified CPC. With 360 hours of data, the *base* (66.4% / 68.3%) and *medium* (66.6% / 68.9%) achieved comparable result with Modified CPC (- / 68.9%) that has a network depth of 6 layers, and as the size of TERA increases, TERA representations achieve better separability and yield better performance than Modified CPC. In the case where 100/360 hours of pre-training data is used, a larger model tends to show performance benefit. As for the case where 960 hours of pre-training data is used, model size tends to be irrelevant and all the TERA models achieved similar performance. On the other hand, we also observe a correlation between performance and data size. As we increase the amount of pre-training data, TERA usually yields better performance.

B. Evaluating Learned Speaker Identity

We presented speaker recognition results in Table III in terms of accuracy. We show results of TERA pre-trained with different combinations of objectives, and with different amount of pre-training data. Here TERA is used as a representation extractor (from the last layer) where parameters are frozen during adapting to speaker classification tasks. We compare the classification accuracy with five baselines: MFCC features, fMLLR features, CPC representations [1], AALBERT representations [21], and Mockingjay representations [20], [21]. MFCC features perform poorly for the task of *Frame* since the MFCC extraction process eliminates most of the pitch information from speech, and only reserves partial tone characteristics. FBANK features perform even worse than MFCC for *Frame*. It shows that the speaker information encoded in FBANK is not linearly accessible. The fMLLR feature extraction process eliminates nearly all of the speaker information and yields an accuracy of 0.4% for *Frame*. The accuracy is similar to a random guess among the 251 speakers ($1/251 \approx 0.003984$). On the other hand, when evaluating fMLLR and FBANK with *Utterance*, a slight improvement is observed, but the performances are still terrible, while MFCC does not improve for *Utterance*.

In contrast to these surface features, TERA recovers the speaker information through its pre-training, even though TERA is pre-trained on the poor performing fMLLR features. When only using the *time* objective, TERA is equivalent to Mockingjay [20] (*row 3*), and yields reasonably well except in *Frame* for limited pre-training data (100 hours or 360 hours). By utilizing two or more auxiliary objectives, performance increases dramatically. Applying the *channel* objective during the pre-training (highlighted in grey in the table) significantly improves performance in the challenging scenarios above (more analysis in Section V-F). We will investigate the effect of *channel* alteration with more details in our ablation study in Section V-F. Adding the *magnitude* objective is also helpful. The objective boosts or yields comparable performance. In general, applying all of the three auxiliary objectives gives the best performance, as the model is pre-trained on more complex input patterns. TERA with all the three objectives consistently outperforms CPC [1], AALBERT [21], and Mockingjay [20] in comparable experiment settings. Note that TERA *base*, AALBERT-3L, and Mockingjay are all 3-layer models, while CPC is a 6-layer model.

As we increase the amount of pre-training data, all of the TERA representations yield better performance. The best performance is achieved (99.5% / 99.8%) when the models are pre-trained with 960 hours of data. The improvement is most significant for *time* and *time + mag*. This shows that models trained with these objectives require larger pre-training data to perform well. On the other hand, *time + channel* and *time + channel + mag* maintained their performance for limited pre-training data, and also benefits for a larger pre-training data size. The reason is that these auxiliary objectives are effective in compensating the shortage of pre-training data. Note that by using all of the three auxiliary objectives, TERA trained on 100/360 hours of data achieved comparable results with the ones trained on 960 hours. This again demonstrates that by utilizing multiple auxiliary objectives, models can benefit in limited pre-training data scenarios (100 hours or 360 hours). We also observe that *Utterance* classification is easier than *Frame*, but both classification tasks show the same trend.

Interestingly, after evaluating the same TERA models on phoneme and speaker classification, we find that TERA captures both speech contents and speaker identity well, and good performance can be attained with simple linear classifiers.

C. Evaluating with ASR

We further apply TERA to speech recognition tasks. The ASR results are presented in Table IV in terms of WER. Similar to the previous experiments, we show the results of TERA pre-trained with different combinations of objectives, and with different amount of pre-training data. Here we use the Kaldi [53] *fglarge* lattice for LM rescoring. The LM rescored WER is denoted as Rescore. All ASR models are trained with 100 hours of labels from the *train-clean-100* subset. TERA here is used for representation extraction (from the last layer), i.e., TERA parameters were frozen when adopting the DNN/HMM framework to speech recognition, and TERA representations are fed to the liGRU model as input. Applying all of the three objectives yields the best performance, whereas applying the *channel* objective on top of *time* is not always better than *time* objective only. This is because the *channel* objective mainly helps encode speaker information. However, utilizing all of the three auxiliary objectives can eliminate this disadvantage. Using *time + channel + mag* with 100 hours of pre-training data achieved similar performance (8.32% / 6.01%) to using *time* with 960 hours (8.31% / 5.99%). We see that 100 hours with three alteration objectives have a similar effect as 960 hours of data. This concretes the fact that using all three auxiliary objectives is potentially increasing the amount of pre-training data (or diversity of data) and eventually leads to improvement. As for the case of 960 hours of pre-training data, using different combinations of auxiliary objectives does not improve performance.

Models	Pre-train	Labels	WER	Rescore
Bidir-CPC [6]	960 hr	96 hr	14.96	9.41
Bidir-CPC [6]	8000 hr	96 hr	13.69	8.70
vq-wav2vec [4]	960 hr	960 hr	6.2	-
wav2vec-large [12]	960 hr	100 hr	-	6.92
DeCoAR [12]	960 hr	100 hr	-	6.10
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
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**
liGRU + TERA-xlarge	960 hr	100 hr	8.47	6.03
liGRU + TERA-base (WS)	960 hr	100 hr	8.46	6.05
liGRU + TERA-med. (WS)	960 hr	100 hr	8.39	6.07

TABLE V: **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 ASR word error rates (WER) on the LibriSpeech [45] *test-clean* subset.

D. Speech Representation for ASR Comparison

In this section, we compare TERA with other speech representation learning methods through ASR. All of the TERA models use a combination of *time + channel + mag* alteration as the auxiliary objectives. In Table V, we list results from recent literature, for training ASR models on top of frozen representations without fine-tuning. All of the works report WER and LM rescored WER (denoted as Rescore) on the *test-clean* subset of LibriSpeech [45]. All of the data for pre-training and downstream adaption are from the same dataset as the last subsection, except for one experiment setup in [6] where 8000 hours of data are used for pre-training. We use the Kaldi [53] *fglarge* lattice for LM rescoring in the experiments of *liGRU + TERA* and its variation, as well as *liGRU + MFCC/fMLLR*. Our LM rescoring scripts are available online[1] for reproducibility. For the previous works including the Bidir-CPC [6], wav2vec-large and DeCoAR [12], 4-gram LM was applied for rescoring. We observe that the model sizes of TERA (i.e., *base*, *medium*, *large*, and *xlarge*) have little influence on the ASR performance when TERA is used as an extractor for speech representation. The *base*

Models	Labels	WER	Rescore
Discrete BERT + vq-wav2vec [5]	100 hr	**4.5**	-
Continuous BERT + wav2vec [5]	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 (FT)	100 hr	8.23	5.84
liGRU + TERA-medium (FT)	100 hr	8.22	5.90
liGRU + TERA-large (FT)	100 hr	**8.00**	**5.80**
MLP + TERA-base (FT)	100 hr	8.47	6.24
MLP + TERA-medium (FT)	100 hr	8.02	5.86
MLP + TERA-large (FT)	100 hr	**7.96**	**5.84**
MLP + TERA-base (FT + WS)	100 hr	8.55	6.12
MLP + TERA-medium (FT + WS)	100 hr	8.17	5.93
MLP + TERA-large (FT + WS)	100 hr	8.19	6.04

TABLE VI: **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. All results are pre-trained on LibriSpeech 960 hours, and uses 100 hours of labels from the *train-clean-100* subset if not specified otherwise.

model is sufficient to improve supervised ASR. TERA models constantly outperform recent speech representation learning approaches, including the model that was pre-trained on 8000 hours of data [6]. We also achieved comparable result with model that used 960 hours of label [4]. The DeCoAR approach (6.10%) achieved similar result with ours (6.01%) in the same setting. We also investigate three baseline features of MFCC, FBANK, and fMLLR. We use identical ASR framework and setting of TERA representations for the three features. The fMLLR feature outperforms FBANK, and FBANK outperforms MFCC, which matches the trend for phoneme classification. Our results suggest that TERA yields constant improvement over surface features in the same ASR framework. We also explored the *WS* approach for combining representations from all layers of TERA for ASR. We find that extracting from the last layer yields better performance than *WS*. The reason is that the model fails to learn meaningful weighted sum when solving the hard ASR task. In our later experiment, we will show *WS* helpful for simpler tasks like phoneme classification. We only explore *WS* for *medium* and *base*, because as concluded above the *base* model is sufficient for ASR, and *WS* is shown to be not effective for this task.

E. Speech Pre-training for ASR Comparison

In this section, we compare results of fine-tuning various pre-trained model for ASR. All of the TERA models use a combination of *time + channel + mag* alteration as the auxiliary objective. We summarize the results from previous literature as well as fine-tuning TERA in *liGRU* or *MLP* framework in Table VI. 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]. All the methods investigated here were pre-trained and adapted with 960 and 100 hours data from LibriSpeech, except one that is adapted with the 360 hours subset of LibriSpeech. The Kaldi [53] *fglarge* lattice is again used for LM rescoring in TERA related experiments. In contrast, vq-wav2vec [5] 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 depth of TERA, and the *large* TERA achieved the best WER. The application of TERA to *liGRU* models is identical to the ones in the previous subsection, except that here we update TERA instead of freezing its parameters during adaptation. By comparing the *liGRU* results in Table V and Table VI, we see that fine-tuning TERA substantially outperforms the case when TERA is simply used for extraction of speech representation. The 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%.

The proposed TERA with *liGRU* or *MLP* outperform Masked Pre-trained Encoders [26] adapted on the same 100 hours of labeled data (9.68%). In addition, with only 100 hours of labeled data we achieved comparable performance (7.96%) with Masked Pre-trained Encoders [26] adapted on 360 hours of labeled data (7.83%). The discrete version of BERT + vq-wav2vec [5] uses a two-step pre-training: first a discrete vocabulary of the data is learned from vq-wav2vec [4], and then a standard BERT [29] is trained on these discrete units. With only a single step of pre-training, our methods achieve comparable results to the discrete BERT + vq-wav2vec [5] (4.5%), and outperform the continuous version of BERT + wav2vec [5] (11.8%). We argue that the two-step pre-training is computation-intensive not only during model building, but also at inference. The discrete BERT + vq-wav2vec [5] is built by stacking a standard BERT model [29] of 12 Transformer Encoder layers [27] on top of vq-wav2vec [4], which consists of an 8-layer encoder network and a 12-layer aggregator network (or context network, as described in [1]). In contrast, our *base* model contains only 3 layers of Transformer Encoder layers and achieves similar ASR performance. Our small encoder architecture (3-layer) benefits for less requirement in computational cost, and has potential to run on edge devices during inference for downstream tasks.

We also fine-tune TERA with *MLP* models, and we find similar trend but sometimes higher WER as compared to TERA with *liGRU*. Using a deeper model with *MLP* gives performance benefit, and *large* achieved 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* is outperformed by *liGRU*, however *MLP* has the advantage of 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. In terms of using *WS* of TERA with MLP, only the *base* model improves, and is not effective for other model depth. The main reason is that for this task, it is too hard for the model to learn ASR and meaningful weighted sum at the same time. We will show that *WS* is effective in a simpler task in the next subsection. To conclude for our proposed method, using a deeper model increases ASR performance during fine-tuning.

	Method	Bidirectional Context	Phone Classification		Speaker Recognition		Speech Recognition	
			Linear	1 Hidden	Frame	Utterance	WER	Rescore
a.	MFCC	-	39.7	59.9	17.6	10.8	8.66	6.42
b.	FBANK	-	42.1	46.9	0.6	5.4	8.64	6.34
c.	fMLLR	-	52.6	68.4	0.4	2.6	8.63	6.25
d.	CPC: random [1]	-	27.6	-	1.87	-	-	-
e.	TERA-base: random	-	15.3	4.8	0.4	0.7	16.96	13.68
f.	TERA-base: none	no	57.0	66.2	1.3	12.7	9.67	7.17
g.	TERA-base: mag	no	59.7	69.5	2.3	28.8	9.32	6.93
h.	TERA-base: channel	no	65.0	76.6	96.7	99.0	9.41	6.91
i.	TERA-base: channel + mag	no	64.2	75.7	97.3	99.2	9.33	6.87
j.	TERA-base: time	yes	64.3	76.8	68.4	96.1	8.46	6.12
k.	TERA-base: time + mag	yes	64.1	77.1	70.8	96.1	8.43	6.11
l.	TERA-base: time + channel	yes	65.2	77.4	93.6	98.5	8.57	6.16
m.	TERA-base: time + channel + mag	yes	65.1	77.3	**98.9**	**99.2**	**8.32**	**6.01**
n.	TERA-base: time + channel + mag (WS)	yes	**65.6**	**78.3**	97.5	**99.2**	8.46	6.05
o.	TERA-base: time + channel + mag (MFCC)	yes	61.5	74.2	95.5	98.8	10.84	8.06
p.	TERA-base: time + channel + mag (FBANK)	yes	**68.0**	76.6	**99.8**	**99.9**	11.83	9.43
q.	CPC: scratch [1]	-	74.6	-	98.5	-	-	-
r.	TERA-base: scratch	-	90.1	83.6	0.4	0.5	10.47	7.68
s.	TERA-base: time + channel + mag (FT)	yes	**90.7**	**91.1**	15.3	1.3	**8.23**	**5.84**

TABLE VII: **Ablation Study.** We use phone classification, speaker recognition and ASR results to study the effect of different auxiliary objectives, including comparison of pre-training with unidirectional or bidirectional context, channel objective that helps learn speaker identity (highlighted in grey), and pre-training with different acoustic features other than fMLLR (MFCC and FBANK). All of the models are pre-trained on the LibriSpeech *train-clean-100* subset, and the numbers are testing results in percentage.

F. Ablation Study

We perform ablation study to better understand the performance of multi-target auxiliary objectives adopted in the TERA pre-training. Results are presented in Table VII, where we measured frame-wise phone classification with *Linear* classifier and *1 Hidden* layer classifier, speaker recognition with *Frame*-wise and *Utterance*-wise linear classification, and finally speech recognition with *liGRU* ASR. For each downstream task, the output of the last layer of TERA is used as the extracted representation if not specified otherwise (no *WS* or *FT*). We incorporate the results from previous sections in Table VII *row a, b, c, j, k, l,* and *m* for comparison. All the reported numbers are percentage and evaluated on test set. The TERA models investigated here are the *base* architectures pre-trained with 100 hours of LibriSpeech.

1) Does Pre-training Truly Help?: In *row d* and *e* of Table VII, we first present results of a CPC network [1] and TERA-base network with random parameters for comparison. As expected, in the three downstream evaluations, we see that surface features constantly outperform random parameter networks. In particular, the *1 Hidden* model for phone classification, as well as *Frame*, and *Utterance* models for speaker classification show serious overfitting issues. Overall the downstream classifiers learn almost nothing from the representation generated by random TERA and random CPC [1]. We conclude that without pre-training, model architecture along provides no benefit.

2) Importance of Time Alteration: Here we study how time auxiliary objectives affect performance. We pre-train TERA with different combinations of objectives, and list the results in Table VII *row f* to *m*. Note that in *row e* (labeled as "none"), during pre-train, the model predicts real frames conditioning on real frames. We split these results into two categories: one with the time alteration that allows the networks to encode bidirectional information via denoising auto encoding, and the other without the *time* objective. We observe that auxiliary objectives injecting the time alteration tend to perform well in phoneme classification and ASR, and does not compromise speaker recognition tasks. Pre-training without the *time* objective yields worse results. The application of *channel* objective (*row h* and *j*), can sometimes close the performance gap, but significant degradation can still be observed in the ASR task. The observation suggests that although Transformer Encoders [27] are bidirectional in nature due to its multi-head self-attention, without the time auxiliary objective the model fails to encode proper context and yield sub-optimal performance. We thus deem the *time* alteration objective to be indispensable. In summary, the *time* alteration objective is effective in learning phonetic information, and representation learned based on the objective improves phoneme prediction and speech recognition.

3) Learning Speaker Identity with Channel Alteration: In Table VII, we highlight all the rows that contain the *channel* objective in grey. We find that by employing the *channel* alteration objective, strong speaker recognition result is achieved. Without channel alteration, speaker recognition performance can drop substantially as suggested in *row f* and *g*. The *time* alteration objective provides some performance increment, but there are still gaps, especially for the harder *Frame* classification task (c.f., *row j* and *k*). In addition, we find that employing the *channel* objective does not compromise phoneme classification and ASR performance (c.f., *row l* and *m*). We surmise this phenomenon results from that TERA representations provide more accessibility to both phonetic and speaker information, and maintain the separability between the two types of information. Downstream models can easily learn to extract the task specific information. In summary, the *channel* alteration objective is effective in learning speaker identity, and representation learned based on this objective yields a more accurate speaker prediction.

4) Effect of Different Knowledge Transfer Techniques: We investigate the transfer learning techniques of *WS* and *FT* on the three downstream tasks. Results are presented in Table VII *row n* and *s*. We also report results *base* model trained from scratch for the three tasks (labeled as *scratch* in *row r*). We include the *scratch* result from the CPC [1] literature (labeled as *scratch* in *row q*). The fully supervised and trained from scratch model helps us understand the benefit of pre-training, and serves as an indication for what is achievable with different model architecture. For *WS* on phone classification, performance is improved (65.6% / 78.3%) over the case where we extract representation from the last layer, *row m* (65.1% / 77.3%). However, for speaker recognition and speech recognition, no substantial improvement is observed. The reason that *WS* did not bring improvement for speaker recognition is that speaker information is most present in the last layer. As for the ASR task, it is too hard for the model to learn *WS* and ASR at the same time. On the other hand, *FT* improves the performance of phoneme classification dramatically (90.7% / 91.1%). Comparing *FT* and *scratch*, we see that *FT* provides better results for both *Linear* and *1 Hidden* classifiers, and *scratch* experiences overfitting with the *1 Hidden* classifier. Obviously self-supervised pre-training is beneficial in performance, as fine-tuning TERA not only outperforms the models trained from scratch, but also provides a more stable supervised training process that avoids overfitting. For *FT* on speaker recognition we experience serious overfitting (*row s*). The fully supervised *scratch* model is also not trainable, and results in an even lower score than *FT* (*row r*). The reason is that the Transformer Encoder [27] architecture is not suitable for speaker recognition tasks. This point is verified by the CPC *scratch* model in *row q*, as CPC equipped with CNN was able to obtain strong results when trained from *scratch* (98.5%, note that this score is higher than the CPC representation which achieved 97.4% [1]). For *FT* on ASR, a clear improvement is obtained over simply using TERA for speech representations. To sum up, extracting speech representation from the last layer performs generally well for all tasks, the *WS* approach is suitable for improving phoneme classification results, and the *FT* approach benefits both phoneme classification and ASR, but may suffer from overfitting for tasks like speaker recognition.

5) Learning with Different Acoustic Features: In Table VII, all of the TERA networks are pre-trained with fMLLR features except for *row o* and *p*, where we pre-trained TERA with *MFCC* and *FBANK*, respectively. Here TERA is frozen for representation extraction from the last layer; *WS* and *FT* are both not used. The architecture of TERA and *liGRU* models are identical for all cases. We can see that pre-training with MFCC yields worse performance in all of the downstream tasks when comparing with TERA adopting the same set of objectives and fMLLR features (*row m*). However, the TERA *base* representation from MFCC still surpasses the performance of directly using MFCC and fMLLR features for phone classification and speaker recognition. For speech recognition, representation from MFCC fails to generalize and degrades in performance. On the other hand, pre-training with FBANK yields better performance for phoneme and speaker classification tasks when comparing to pre-training with fMLLR (*row m*). However, for speech recognition, representation from FBANK has even worse results than representation from MFCC. The reason that pre-training with MFCC or FBANK yields worse results for our ASR framework is because of overfitting. Note that the results we obtained here may not apply to all ASR systems, we are not saying that one acoustic feature is superior than another. We conclude that despite the same architecture and training objectives, pre-training with different acoustic features can lead to significantly different results.

Models	Pre-train	PER
CNN + TD-filterbanks [55]	None	18.0
CNN + HMM [56]	None	16.5
liGRU + MFCC [57]	None	16.7
liGRU + FBANK [57]	None	15.8
liGRU + fMLLR [57]	None	14.9
wav2vec [3]	80 hr	17.6
wav2vec [3]	960 hr	15.6
wav2vec [3]	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 (WS)	960 hr	14.6
liGRU + TERA-base (FT)	960 hr	15.2
MLP + TERA-base (FT)	960 hr	16.6
liGRU + TERA-medium	960 hr	14.9

TABLE VIII: **Comparison of pre-training approaches between recent work and the proposed approach on TIMIT [52].** All the pre-training data are from LibriSpeech [45], if not specified otherwise. All of the TERA models use the combined auxiliary objective of time + channel + mag alteration.

G. 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 [52] ASR task. The same Hybrid ASR setting and framework as described above for LibriSpeech ASR is used, except that we adopt a learning rate of $4e^{-4}$ and a batch size of 8. Testing results of TERA and another self-supervised learning technique, wav2vec [3], are summarized in Table VIII in terms of PER. We also list the results of strong supervised systems [55]–[57]. All of the TERA models use a combination of *time* + *channel* + *mag* alteration as the auxiliary objectives, and are pre-trained with various amount of data. As expected, pre-training on 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, using either *WS* or *FT* is not helpful, and extracting speech representations from the last layer provides the best performance. Also, there is no significant gain when extracting features from a larger model *medium*. The reason is likely because there is not enough labeled data in TIMIT. On the other hand, our best model (14.5%) outperforms wav2vec [3] (14.7%) that uses both LibriSpeech [45] and Wall Street Journal (WSJ) data for pre-training, as well as all the strong supervised baselines. By using only 100 hours of pre-training data, we even surpass the result of wav2vec that uses 960 hours for pre-training (15.2% v.s. 15.6%).

VI. Conclusion

We propose a novel multi-target self-supervised training scheme called TERA, where we use multiple auxiliary objectives instead of one during pre-training. 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 tasks of phone classification, speaker recognition, and speech recognition. We conduct a complete ablation study, and a thorough comparison on recent representation learning and pre-training approaches. In our analysis, we find that the proposed objectives allow TERA to encode phonetic and speaker information in a linearly separable way. With the proposed alterations, the diversity of input is increased, and the performance, especially for the case where pre-training data is limited, is improved. We also explore various knowledge transfer approaches to incorporate the pre-trained model in the downstream tasks, and we find each of the approach is suitable for different tasks. The choice of acoustic features also plays a crucial role in the reconstruction-based self-supervised learning, as different surface features will lead to significantly different downstream performance. Furthermore, in an ASR task, we show that TERA pre-trained on one dataset can be easily transferred to another, and outperforms recent approaches. We show that TERA brings improvement for downstream tasks, especially those with limited training data, and benefit downstream tasks where it is expensive to collect training data. In future work we will investigate and deploy TERA in more downstream tasks including voice conversion, speech denoising, speech separation, speech translation, and speech QA.

References

[1] A. van den Oord, Y. Li, and O. Vinyals, "Representation learning with contrastive predictive coding," 2018.

[2] J. Ebbers, M. Kuhlmann, and R. Haeb-Umbach, "Adversarial contrastive predictive coding for unsupervised learning of disentangled representations," 2020.

[3] S. Schneider, A. Baevski, R. Collobert, and M. Auli, "wav2vec: Unsupervised pre-training for speech recognition," *Interspeech*, 2019.

[4] A. Baevski, S. Schneider, and M. Auli, "vq-wav2vec: Self-supervised learning of discrete speech representations," *arXiv preprint arXiv:1910.05453*, 2019.

[5] A. Baevski, M. Auli, and A. Mohamed, "Effectiveness of self-supervised pre-training for speech recognition," 2019.

[6] K. Kawakami, L. Wang, C. Dyer, P. Blunsom, and A. van den Oord, "Learning robust and multilingual speech representations," 2020.

[7] M. Rivière, A. Joulin, P.-E. Mazaré, and E. Dupoux, "Unsupervised pretraining transfers well across languages," 2020.

[8] Y.-A. Chung, W.-N. Hsu, H. Tang, and J. Glass, "An unsupervised autoregressive model for speech representation learning," in *Interspeech*, 2019.

[9] Y.-A. Chung and J. Glass, "Generative pre-training for speech with autoregressive predictive coding," in *ICASSP*, 2020.

[10] ——, "Improved speech representations with multi-target autoregressive predictive coding," 2020.

[11] Y.-A. Chung, H. Tang, and J. Glass, "Vector-quantized autoregressive predictive coding," 2020.

[12] S. Ling, Y. Liu, J. Salazar, and K. Kirchhoff, "Deep contextualized acoustic representations for semi-supervised speech recognition," 2019.

[13] M. Tagliasacchi, B. Gfeller, F. de Chaumont Quitry, and D. Roblek, "Self-supervised audio representation learning for mobile devices," 2019.

[14] M. Tagliasacchi, B. Gfeller, F. d. C. Quitry, and D. Roblek, "Pre-training audio representations with self-supervision," *IEEE Signal Processing Letters*, 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 Transactions on Audio, Speech, and Language Processing*, vol. 27, no. 12, p. 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," *Interspeech*, Sep 2019.

[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," *Interspeech 2019*, Sep 2019.

[19] S. Khurana, A. Laurent, W.-N. Hsu, J. Chorowski, A. Lancucki, R. Marxer, and J. Glass, "A convolutional deep markov model for unsupervised speech representation learning," 2020.

[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," *ICASSP 2020*, May 2020.

[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," 2020.

[22] W. Wang, Q. Tang, and K. Livescu, "Unsupervised pre-training of bidirectional speech encoders via masked reconstruction," *ICASSP 2020*, May 2020.

[23] X. Song, G. Wang, Z. Wu, Y. Huang, D. Su, D. Yu, and H. Meng, "Speech-xlnet: Unsupervised acoustic model pretraining for self-attention networks," 2019.

[24] D. Jiang, X. Lei, W. Li, N. Luo, Y. Hu, W. Zou, and X. Li, "Improving transformer-based speech recognition using unsupervised pre-training," 2019.

[25] D. Jiang, W. Li, R. Zhang, M. Cao, N. Luo, Y. Han, W. Zou, and X. Li, "A further study of unsupervised pre-training for transformer based speech recognition," 2020.

[26] L. Liu and Y. Huang, "Masked pre-trained encoder base on joint ctc-transformer," 2020.

[27] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," 2017.

[28] A. T. Liu and Y. Shu-wen, "S3PRL: The self-supervised speech pre-training and representation learning toolkit," 2020. [Online]. Available: https://github.com/andi611/Self-Supervised-Speech-Pretraining-and-Representation-Learning

[29] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," 2018.

[30] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, and V. Stoyanov, "Roberta: A robustly optimized bert pretraining approach," 2019.

[31] A. Graves, S. Fernández, F. Gomez, and J. Schmidhuber, "Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks," in *Proceedings of the 23rd international conference on Machine learning*, 2006, pp. 369–376.

[32] T. Mikolov, M. Karafiát, L. Burget, J. Černockỳ, and S. Khudanpur, "Recurrent neural network based language model," in *Eleventh annual conference of the international speech communication association*, 2010.

[33] M. Peters, M. Neumann, M. Iyyer, M. Gardner, C. Clark, K. Lee, and L. Zettlemoyer, "Deep contextualized word representations," *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, 2018.

[34] Z. Lan, M. Chen, S. Goodman, K. Gimpel, P. Sharma, and R. Soricut, "Albert: A lite bert for self-supervised learning of language representations," 2019.

[35] Z. Yang, Z. Dai, Y. Yang, J. Carbonell, R. Salakhutdinov, and Q. V. Le, "Xlnet: Generalized autoregressive pretraining for language understanding," 2019.

[36] H. Wu, A. T. Liu, and H. yi Lee, "Defense for black-box attacks on anti-spoofing models by self-supervised learning," 2020.

[37] S. wen Yang, A. T. Liu, and H. yi Lee, "Understanding self-attention of self-supervised audio transformers," 2020.

[38] D. S. Park, W. Chan, Y. Zhang, C.-C. Chiu, B. Zoph, E. D. Cubuk, and Q. V. Le, "Specaugment: A simple data augmentation method for automatic speech recognition," *Interspeech 2019*, Sep 2019.

[39] P. Wang, L. Wei, Y. Cao, J. Xie, and Z. Nie, "Large-scale unsupervised pre-training for end-to-end spoken language understanding," in *ICASSP*, 2020.

[40] S. Ling, J. Salazar, Y. Liu, and K. Kirchhoff, “BERTphone: Phonetically-aware Encoder Representations for Utterance-level Speaker and Language Recognition,” in *Proc. Odyssey 2020 The Speaker and Language Recognition Workshop*, 2020.
[41] T. Mikolov, K. Chen, G. Corrado, and J. Dean, “Efficient estimation of word representations in vector space,” 2013.
[42] J. Howard and S. Ruder, “Universal language model fine-tuning for text classification,” 2018.
[43] C. Sun, X. Qiu, Y. Xu, and X. Huang, “How to fine-tune bert for text classification?” *Chinese Computational Linguistics*, p. 194–206, 2019.
[44] A. Chronopoulou, C. Baziotis, and A. Potamianos, “An embarrassingly simple approach for transfer learning from pretrained language models,” *Proceedings of the 2019 Conference of the North*, 2019.
[45] V. Panayotov, G. Chen, D. Povey, and S. Khudanpur, “Librispeech: An ASR corpus based on public domain audio books,” in *ICASSP*, 2015.
[46] N.-Q. Pham, T.-S. Nguyen, J. Niehues, M. Müller, S. Stüker, and A. Waibel, “Very deep self-attention networks for end-to-end speech recognition,” 2019.
[47] C.-Y. Li, P.-C. Yuan, and H.-Y. Lee, “What does a network layer hear? analyzing hidden representations of end-to-end asr through speech synthesis,” *ICASSP 2020*, May 2020.
[48] D. P. Kingma and J. Ba, “Adam: A method for stochastic optimization,” 2014.
[49] J. Yosinski, J. Clune, Y. Bengio, and H. Lipson, “How transferable are features in deep neural networks?” 2014.
[50] N. F. Liu, M. Gardner, Y. Belinkov, M. E. Peters, and N. A. Smith, “Linguistic knowledge and transferability of contextual representations,” *Proceedings of the 2019 Conference of the North*, 2019.
[51] M. Ravanelli, T. Parcollet, and Y. Bengio, “The pytorch-kaldi speech recognition toolkit,” in *In Proc. of ICASSP*, 2019.
[52] 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 technical report n*, vol. 93, 1993.
[53] D. Povey, A. Ghoshal, G. Boulianne, L. Burget, O. Glembek, N. Goel, M. Hannemann, P. Motlicek, Y. Qian, P. Schwarz, J. Silovsky, G. Stemmer, and K. Vesely, “The kaldi speech recognition toolkit,” in *ASRU*, 2011.
[54] K.-F. Lee and H.-W. Hon, “Speaker-independent phone recognition using hidden markov models,” *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 37, no. 11, pp. 1641–1648, 1989.
[55] N. Zeghidour, N. Usunier, I. Kokkinos, T. Schaiz, G. Synnaeve, and E. Dupoux, “Learning filterbanks from raw speech for phone recognition,” *ICASSP 2018*, Apr 2018.
[56] L. Tóth, “Phone recognition with hierarchical convolutional deep maxout networks,” *EURASIP Journal on Audio, Speech, and Music Processing*, vol. 2015, no. 1, pp. 1–13, 2015.
[57] M. Ravanelli, P. Brakel, M. Omologo, and Y. Bengio, “Light gated recurrent units for speech recognition,” *IEEE Transactions on Emerging Topics in Computational Intelligence*, vol. 2, no. 2, p. 92–102, Apr 2018.

Andy T. Liu received his Bachelor’s degree in Electrical Engineering (EE) from National Taiwan University (NTU), Taipei, Taiwan in 2018. He is currently a Ph.D. student at the Graduate Institute of Communication Engineering (GICE) at National Taiwan University. He mainly works on self-supervised and unsupervised algorithms of speech, including speech pre-training, representation learning, and voice conversion.

Shang-Wen Li is a Senior Applied Scientist at Amazon AWS AI since 2019. He worked at Apple Siri and Amazon Alexa before joining AWS. He earned his Ph.D. from MIT Computer Science and Artificial Intelligence Laboratory (CSAIL) in 2016 supervised by Professor Victor Zue. His research is focused on 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 (NTU), Taipei, Taiwan, in 2010 and 2012, respectively. From September 2012 to August 2013, he was a postdoctoral fellow in Research Center for Information Technology Innovation, Academia Sinica. From September 2013 to July 2014, he was a visiting scientist at the Spoken Language Systems Group of MIT Computer Science and Artificial Intelligence Laboratory (CSAIL). He is currently an assistant professor of the Department of Electrical Engineering of National Taiwan University, with a joint appointment at the Department of Computer Science & Information Engineering of the university. His research focuses on spoken language understanding, speech recognition and machine learning.