Slowness Regularized Contrastive Predictive Coding for Acoustic Unit Discovery

Saurabhchand Bhati, Jesús Villalba, Piotr Żelasko, *Member, IEEE*, Laureano Moro-Velazquez, *Member, IEEE*, and Najim Dehak, *Senior Member, IEEE*

***Abstract*—Self-supervised methods such as Contrastive predictive Coding (CPC) have greatly improved the quality of the unsupervised representations. These representations significantly reduce the amount of labeled data needed for downstream task performance, such as automatic speech recognition. CPC learns representations by learning to predict future frames given current frames. Based on the observation that the acoustic information, e.g., phones, changes slower than the feature extraction rate in CPC, we propose regularization techniques that impose slowness constraints on the features. Here we propose two regularization techniques: Self-expressing constraint and Left-or-Right regularization. We evaluate the proposed model on ABX and linear phone classification tasks, acoustic unit discovery, and automatic speech recognition. The regularized CPC trained on 100 hours of unlabeled data matches the performance of the baseline CPC trained on 360 hours of unlabeled data. We also show that our regularization techniques are complementary to data augmentation and can further boost the system's performance. In monolingual, cross-lingual, or multilingual settings, with/without data augmentation, regardless of the amount of data used for training, our regularized models outperformed the baseline CPC models on the ABX task.**

***Index Terms*—Contrastive predictive coding, self-supervised learning, unsupervised learning, zero resource speech processing.**

I. Introduction

THE speech signal contains information about linguistic units [1], speaker identity [2], the emotion of the speaker [3], etc. In a supervised scenario, the manual labels guide a strong classifier, such as a Deep Neural Network (DNN), to extract task-specific features. For example, paired with phone labels, a DNN learns to focus on extracting the acoustic information from speech and suppress the information about the speaker's identity. When paired with speaker labels, the DNN learns to focus on speaker information and suppress the phone information.

Received 30 May 2023; revised 23 November 2023; accepted 19 December 2023. Date of publication 15 January 2024; date of current version 30 September 2024. This work was supported by NSF under Grant 1909075. The associate editor coordinating the review of this article and approving it for publication was Prof. Joseph Keshet. *(Corresponding author: Saurabhchand Bhati.)*

Saurabhchand Bhati and Laureano Moro-Velazquez are with the Center for Language, Speech Processing, Johns Hopkins University, Baltimore, MD 21218 USA (e-mail: sbhati1@jhu.edu; laureano@jhu.edu).

Jesús Villalba and Najim Dehak are with the Center for Language, Speech Processing, Johns Hopkins University, Baltimore, MD 21218 USA, and also with the Human Language Technology Center of Excellence, Johns Hopkins University, Baltimore, MD 21218 USA (e-mail: jvillalba@jhu.edu; ndehak3@jhu.edu).

Piotr Żelasko is with the Meaning.Team Inc., Johns Hopkins University, Baltimore, MD 21218 USA (e-mail: pzelasko@meaning.team).

Digital Object Identifier 10.1109/TASLP.2024.3350888

In the unsupervised scenario, we do not have the guidance of manual transcriptions to select the relevant features and marginalize irrelevant information. A speech representation that captures the underlying relevant information becomes crucial for unsupervised systems' good performance [4], [5], [6], [7], [8], [9], [10], [11], [12], [13]. Learning such representations from unlabeled speech data could enable speech technologies in low-resource languages where limited or no amounts of labeled data are available [4], [5], [6], [7], [8], [9], [10], [11], [12], [13]. The goal of unsupervised representation learning is to capture the phone information and disentangle the other sources of information, such as the speaker or channel.

Self-supervised learning (SSL) methods have emerged as a promising technique for representation learning from unlabeled speech data [14], [15], [16], [17]. SSL has also been shown to be effective for learning representations in natural language processing [18], [19] and computer vision [20], [21]. SSL methods learn representation by solving an auxiliary task for which the labels can be generated from the unlabeled data. For example, in Contrastive Predictive Coding (CPC) [14], a popular self-supervised method, the auxiliary task is next frame prediction. A CNN learns representations from the raw waveform, which are then fed into a recurrent neural network to generate contextual representations. The model is trained via noise contrastive estimation to correctly identify the correct next frame from a set of random frames given the contextual features.

Self-supervised techniques such as CPC have drastically improved the quality of the representation learned from unlabeled data [14], [22]. CPC extracts a feature vector every 10ms, i.e., 100 features/second, whereas underlying information, e.g., phones, change much more slowly. There have been several solutions to impose slow changes to the latent representations [23], [24], [25], [26], [27]. In this work; we propose regularizing constraints to impose slow changes in the latent representations. Ideally, the representation would stay constant within a phone and change abruptly at the phone boundaries.

Self-expressing autoencoders (SEA) [28] add an extra self-expressing constraint as a regularization term to the autoencoder. SEA tries to express the features extracted from the encoder as a linear combination of other features, thus enforcing the underlying information is shared among features. We modify the self-expressing constraint and use it to regularize the CPC training. We also propose Left-or-Right regularization (LorR) to constrain the nearby frames to be similar. LorR assumes a given frame shares a phone label with either the left or right frames.

We add extra loss that minimizes the variance between the given frame and adjacent frames.

We pretrained the baseline CPC model and our proposed regularized CPC on Librispeech 100 hours and 360 hours portions. We evaluate the models on the ABX task of the Zerospeech 2017 benchmark, acoustic unit discovery, and the linear phone classification task on Librispeech 100 hours. We carry out a detailed hyper-parameter search to find the optimal weights for adding the regularization loss with the CPC loss. Experiments show that we outperform the baseline CPC on both the ABX and the linear phone classification tasks.

We also train CPC models on English, French, and Mandarin datasets from the Zerospeech 2017 challenge and evaluate the performance in monolingual, cross-lingual, and multilingual settings. We also evaluate the CPC models on ASR across ten languages from the common voice dataset. Across all these conditions, our regularization consistently improves the system's performance.

Data augmentation has become an important part of both supervised [29] and self-supervised systems [20], [21], [30]. It allows us to train larger models by reducing overfitting. One major direction for SSL models is to learn augmentation invariant representation where a model is presented with two different augmented versions, and it must generate similar representations [20], [21]. Data augmentation combined with CPC significantly improves over baseline CPC [30]. In this approach, the past and/or future audio signals are augmented with different augmentations, and the model is trained to predict the correct next frame. We show that our regularization techniques are complementary to the data augmentation techniques and can be used on top to improve the system performance further.

The contributions of this work are summarized below:

- We propose Left-or-Right regularization and Self-expression to enforce slowness constraints on the CPC features
- We show the proposed regularizations improve performance on ABX, linear phone classifier, acoustic unit discovery, and automatic speech recognition tasks
- We show our proposed regularization's work in monolingual, cross-lingual, and multilingual conditions
- We show that the proposed regularizations are complementary and can be used with augmentation to improve the system's performance further

II. Related Work

In this work, we focus on unsupervised feature learning, where we do not have any labels for the training data. The ZeroSpeech challenges [31], [32], [33], [34], [35] have been some of the significant drivers of progress in the unsupervised feature learning area. Same-different or ABX tasks have consistently been part of all the challenges that focus on evaluating the representations' quality. Before the popularity of SSL techniques, autoencoders [23], [28], [36], [37] were a dominant paradigm for learning representations. Autoencoders consist of two parts: an encoder which maps an input to a latent space, and a decoder which tries to reconstruct the input. The autoencoders are optimized to minimize the difference between the original and reconstructed inputs. Variational autoencoders (VAE) [38] proposed a different probabilistic interpretation of the feature learning framework. Vector Quantized VAE [39] replaced the continuous and stochastic latent vectors with deterministically quantized vectors. Since the quantization is not differentiable, a straight-through estimator is used to optimize the codebook used for quantization. Chorowski et al. [23] proposed a wavenet-autoencoder that encodes MFCC features into a latent space via a VAE and uses a wavenet decoder to reconstruct the original waveform.

Contrastive Predictive Coding (CPC) [14] and its variants [17], [24], [25], [26], [27], [40], [41] have emerged as a popular choice for the representation learning task. Some of the best-performing solutions to previous challenges are CPC-based [25], [41]. Even though some autoencoder-based methods tend to be more data efficient than CPC, given more data, CPC outperforms them on the ABX task [17]. For the recent zero speech challenge, CPC has been the choice of feature extractor [35]. However, quantizing the representations from the CPC degrades the performance on the ABX task [35]. We chose the CPC as one of the baselines in the present work.

There have been several attempts to impose slow changes on the unsupervised representations extracted from speech data. Slow-feature analysis [42] imposed a penalty on the rate of change of features to encourage slow changes in the features. A time-jitter regularization [23] was proposed to reduce the variability between adjacent embeddings of VQ-VAE. Chorowski et al. [43] added a penalty to divide the VQ-VAE features into a given number of piecewise-constant pieces. Although, this requires knowing the number of segments in advance. Kamper et al. [44] proposed a dynamic programming-based generalization of this approach to obtain phone segmentation from VQ-VAE and VQ-CPC features. Kamper et al. [45] further extended this idea to apply dynamic programming (DP) iteratively to perform phone and word segmentation. The first step performs bottom-up phone discovery using DP and then performs symbolic word segmentation on top of the discovered units. The two stages are trained separately, i.e., word segmentation does not influence phone discovery.

Bhati et al. [24] proposed Segmental CPC (SCPC): a hierarchical model which stacked two CPC modules operating at different time scales. The lower CPC operates at the frame level, and the higher CPC operates at the phone-like segment level. A simple differentiable boundary detector generates phone-like segments used for training the segment-level CPC. Both lower and higher levels CPCs are optimized jointly. They demonstrated that adding the second level CPC improves the phone boundary detection but degrades the phone class information present in the learned features [46]. Chorowski et al. [25] proposed Aligned CPC (ACPC), in which the model outputs a sequence of $K < M$ predictions that are aligned to the M upcoming representations. mACPC [26] proposed a hierarchical model similar to SCPC where they used ACPC as the building blocks instead of CPC. mACPC obtained better feature discrimination than CPC, SCPC, and ACPC. mACPC further confirmed the tradeoff between boundary detection and classification performance.

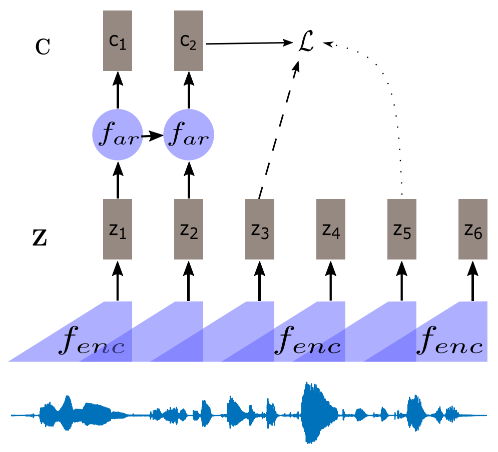

Fig. 1. Overview of the CPC architecture. The solid line represents the reference frame; the dashed line shows the positive, and the dotted line shows the randomly sampled negative example.

Hierarchical CPC (HCPC) [27] stacked two CPC models and used reinforcement learning to generate the segment boundaries. They showed that it is possible to improve the classification performance of the features extracted from multilevel CPC by training the second-level CPC on segments extracted to optimize the next segment prediction task directly. There still seems to be some tradeoff between prediction quality and segmentation performance. HCPC obtains better phone discrimination than SCPC and mACPC but has lower phone segmentation performance than SCPC and mACPC. HCPC achieves state-of-art performance on the Zerospeech 2021 task.

ACPC [25], mACPC [26] and HCPC [27] all obtain better performance on ABX task than the baseline CPC. We compare our proposed regularization methods with all of them and show that we outperform them on the ABX task.

III. Regularized Contrastive Predictive Coding

A. Contrastive Predictive Coding

Contrastive Predictive Coding (CPC) [14] learns representations by predicting future feature frames from past frames. The architecture is shown in Fig. 1. CPC can learn representations directly from raw speech waveforms. A convolution encoder, $f_{\text{enc}}: \mathbf{X} \rightarrow \mathbf{Z}$, maps the audio waveform, $\mathbf{X}$ to latent spectral representations, $\mathbf{Z}(\in \mathbb{R}^{d \times L}) = (\mathbf{z}_1, \mathbf{z}_2, \ldots, \mathbf{z}_L)$. In the most common setting, each d-dimensional vector $\mathbf{z}_i$ corresponds to a 30ms audio frame extracted with a 10ms shift. A recurrent neural network, $f_{\text{ar}}: \mathbf{Z} \rightarrow \mathbf{C}$, extracts contextual representations $(\mathbf{c}_1, \mathbf{c}_2, \ldots, \mathbf{c}_L)$ computed as $\mathbf{c}_i = f_{\text{ar}}(\mathbf{z}_i)$. Given a reference context representation $\mathbf{c}_t$ the model needs to identify the next frame $\mathbf{z}_{t+m}$ correctly from a set, $\mathcal{Z}_t$, of $K+1$ representations, which includes $\mathbf{z}_{t+m}$ and K distractors. W_m is the linear transformation used for predicting $\mathbf{z}_{t+m}$ from $\mathbf{c}_t$. The overall loss is given as follows:

$$\mathcal{L}_{\text{CPC}} = -\frac{1}{M} \sum_{m=1}^{M} \log \frac{\exp(\mathbf{z}_{t+m}^T W_m \mathbf{c}_t)}{\sum_{\tilde{\mathbf{z}} \in \mathcal{Z}_t} \exp(\tilde{\mathbf{z}}^T W_m \mathbf{c}_t)} \tag{1}$$

B. Self-Expressing Autoencoders

In an autoencoder, an encoder, enc maps the input $\mathbf{X}$ into a latent space $\mathbf{Z} = \text{enc}(\mathbf{X})$ and the decoder dec takes the $\mathbf{Z}$ and tries to reconstruct the input from it. The autoencoder is trained to minimize the reconstruction loss: $\|\mathbf{X} - \text{dec}(\mathbf{Z})\|_2^2$.

The self-expressing autoencoders (SEA) [28] introduce self-expressing constraints to encourage autoencoders to learn representations highlighting underlying phone information. The self-expressing constraint tries to make the embeddings of frames that belong to the same distribution to be as similar as possible and frames that belong to different distributions as dissimilar as possible.

In SEA, an encoder maps the input features into an embedding space, and two decoders with shared parameters try to reconstruct the input as well as possible. One decoder tries to reconstruct the input from the embedding, and another tries to reconstruct from their self-expressed version.

To compute the self-expressed version of the encoder outputs, we first compute the affinity matrix $\mathbf{A}$, which captures the pair-wise cosine similarities between frames, $\mathbf{A}_{ij}$ denotes the cosine similarity between features $\mathbf{z}_i$ and $\mathbf{z}_j$. The last layer of the encoder is ReLU nonlinearly, and thus the $\mathbf{z}$ is non-negative, and therefore $\mathbf{A}$ is non-negative. We then remove the contribution of the diagonal entries from $\mathbf{A}$ and normalize each row, to sum up, to $\mathbf{1}$. The self-expressed version, $\underline{\mathbf{Z}}$, is given as

$$\underline{\mathbf{Z}} = \text{rownorm}(\mathbf{A} - \mathbf{I})\mathbf{Z} \tag{2}$$

where rownorm denotes the row normalization operation. The SEA is trained to minimize the reconstruction loss

$$\mathcal{L}_{\text{SE}} = \|\mathbf{X} - \text{dec}(\mathbf{Z})\|_2^2 + \|\mathbf{X} - \text{dec}(\underline{\mathbf{Z}})\|_2^2 \tag{3}$$

For the i^{th} input, $\mathbf{x}_i$, to minimize the reconstruction error, the decoder needs to reconstruct $\mathbf{x}_i$ from both $\mathbf{z}_i$ and $\underline{\mathbf{z}}_i$. $\underline{\mathbf{z}}_i$ is a linear combination of features except for $\mathbf{z}_i$. Other features should have the information present in $\mathbf{z}_i$. Thus the information present in $\mathbf{z}_i$ must be shared in other features.

C. CPC With Self-Expressive Constraint

In the original SEA [28], to ensure the self-similarity matrix, $\mathbf{A}$, is non-negative, the last layer in the encoder is ReLU nonlinearity which restricts $\mathbf{z}$ to non-negative values. In the CPC architecture, the last layer of the encoder is ReLU as well. We calculate $\underline{\mathbf{Z}}$ using (2).

The CPC model does not use a decoder to learn representations. So, instead of minimizing the difference between input and reconstructed input from the self-expressed embeddings, we force the embeddings and their self-expressed versions to be as close as possible. The total loss of the model is given as

$$\mathcal{L}_{\text{CPC+SE(ReLU)}} = \mathcal{L}_{\text{CPC}} + \lambda \|\mathbf{Z} - \underline{\mathbf{Z}}\|_2^2 , \tag{4}$$

where λ is the regularization weight.

D. ***L***eft-***or***-***R***ight (LorR) Regularization

Assuming a phone consists of at least two feature frames, then any feature frame would have the same phone label as either the left or right frame. Most of the time, both left and right frames would have the same phone label. Only at the phone boundaries, which are much fewer than the total number of frames, the phone labels for left and right frames would be different. With the 30ms context size and 10ms shift for the convolutional feature extractor, the minimum two frames phone assumption works out to be 40ms. Let's consider a sequence of four feature frames $\mathbf{z}_{i-1}, \mathbf{z}_i, \mathbf{z}_{i+1}, \mathbf{z}_{i+2}$. We want to constrain the features from the same phone to be as close as possible. However, in unsupervised scenarios, we do not know the phone labels. We minimize the minimum of variance between $\mathbf{z}_{i-1}, \mathbf{z}_i$ and $\mathbf{z}_i, \mathbf{z}_{i+1}$. At boundaries having a minimum allows the loss to be flexible, and it can choose the side with minimum variance, e.g., if there is a boundary at $\mathbf{z}_i$, then the model could pick and minimize the variance between $\mathbf{z}_i, \mathbf{z}_{i+1}$ and vice-versa. In the middle of a phone, when both the left and right sides belong to the phone, the choice of side does not matter. We can extend this idea and try to enforce w frames to be similar. The loss at i_{th} feature is given as:

$$\mathcal{L}_i = \min\left(\sum_d \text{Var}(\mathbf{z}_{i-w+1:i}), \sum_d \text{Var}(\mathbf{z}_{i+1:i+w})\right) \quad (5)$$

Where $\sum_d$ denotes the sum of element-wise variance across d dimensions. The total LorR loss is average across all the time indexes

$$\mathcal{L}_{\text{LorR}} = \frac{1}{L}\sum_{i=0}^{L}\mathcal{L}_i \quad (6)$$

The regularized CPC is optimized to minimize both the CPC loss and the LorR loss.

$$\mathcal{L}_{\text{CPC+LorR}} = \mathcal{L}_{\text{CPC}} + \alpha\mathcal{L}_{\text{LorR}} \quad (7)$$

where α is the regularization weight.

IV. Experiments

A. Tasks and Datasets

We used LibriSpeech 100 hours, 360 hours, and the Zerospeech 2017 datasets for pretraining the CPC and regularized version of CPC. The Zerospeech 2017 train subset contains 45, 24, and 2.5 hours of data across English, French, and Mandarin, respectively. One common task for evaluating the quality of the representations is probing for the phone information by training a linear phone classifier. For the supervised training of the linear phone classifier, we used the train/test splits and force alignments for Librispeech-100h from [17]. We also evaluated how well these representations can be clustered and mapped into discrete symbols.

Another common task is the ABX phone discrimination task [47]. ABX task measures the phone separability of the representations obtained from the feature extractor. Features from two instances of the same phone should be closer than two instances of different phones. For example, if phone instance a and x belong to the same phone class A and phone instance b belongs to phone class B, then $d(a,x) < d(b,x)$ where d is some distance metric. The ABX task was done in two modes: within speaker–when a, b, x belong to the same speaker– and across speaker–when a, b belong to the same speaker, but x belongs to a different speaker. For the ABX and linear phone classification task, we used the implementation provided by [17]. ABX task does not require training any additional components to evaluate the quality of the learned representations.

For the ABX task, we used the Zerospeech 2017 and Zerospeech 2021 challenge datasets for evaluations. Zerospeech 2017 test set contains the same speakers from the train set and an unknown number of new speakers appearing in 1-second, 10 seconds, and 120 seconds files. This allows us to measure the speaker invariance of the feature extraction system and the impact of utterance length on the learned representations. More details about the dataset and the evaluation can be found in the challenge paper [33]. The Zerospeech 2021 uses the standard Librispeech validation and test splits as validation test splits.

For the ABX task, we also trained the baseline and the regularized CPC models on English, French, and Mandarin datasets. This allows us to evaluate our models across different languages with varying training sizes in a monolingual setting. We also train multilingual systems with data pooled from all three languages together. For the cross-lingual setting, we used the CPC models trained on Librispeech. We evaluated the systems across the three languages for the multilingual and cross-lingual settings.

B. Architecture Details

We followed the improved CPC [17], which replaces the batch-norm with channel-norm. This helps stabilize the model training and prevents poor solutions. Each of the linear transformation, W_m, used for predicting $\mathbf{z}_{t+m}$ from $\mathbf{c}_t$ is now replaced with a single-layer transformer [48]. This new modified layer can access the entire past till time t to predict $\mathbf{z}_{t+k}$. Dropout is used in the transformer layers to improve the system's performance. Gated recurrent unit (GRU) is replaced with LSTM as the recurrent neural network for generating contextual representations. This modification improves the downstream performance of the features extracted from the CPC. These modifications allow us to reduce the number of channels in the convolutions layer from 512 to 256 without impacting the performance while significantly reducing the memory requirements.

In CPC, the encoder was a 5-layer convolutional network with 256 channels in each layer with kernel sizes: 10,8,4,4,4 and strides 5,4,2,2,2. The encoder had a total downsampling factor of 160, with a stride of about 10ms. For a 16 kHz input, each feature encodes 10ms of audio and generates 100 features per second. We used a single-layer LSTM with 256 hidden units as the recurrent network. We predict 12 frames into the future, i.e., M = 12, and use 128 negative examples. All the models are trained with a batch size of 12 on a single GPU for 100 epochs.

C. ABX and Linear Phone Classification

We evaluated the proposed algorithms on ABX and the phone classification tasks. Table I shows the performance of features extracted from baseline CPC and CPC models trained with various regularization techniques on the Librispeech 100 hours

TABLE I
ABX Scores on ZS17 English Dataset

	Within				Across			
	1s	10s	120s	avg	1s	10s	120s	avg
CPC	7.3	7.1	7.2	7.2	10.3	9.6	9.8	9.9
CPC + SE (ReLU)	6.6	6.5	6.8	6.6	9.4	8.9	9.2	9.2
CPC + LorR	5.8	5.6	6.1	5.9	8.4	8.0	8.5	8.3
CPC + SE (ReLU) + LorR	6.2	5.8	6.1	6.0	8.8	8.3	8.5	8.5
CPC-2L + LorR	**5.6**	**5.2**	**5.5**	**5.5**	**8.5**	**7.8**	**8.1**	**8.1**

CPC models are trained on librispeech 100h portion.

TABLE II
ABX Performance on ZS17 English Dataset, the CPC Model is Pretrained on the Librispeech 360 Dataset

	Within	Across
Trained on Zerospeech2017 (45h)		
Supervised topline [33]	5.3	6.9
Heck et al. [48]	6.2	8.7
Chorowski et al. [23]	5.5	8.0
CPC	9.4	13.0
CPC+LorR	7.6	11.9
Trained on Librispeech 360		
Original CPC [14]	9.6	13.0
CPC [17]	6.5	8.5
CPC+LorR	5.5	7.4
CPC-2L+LorR	**4.7**	**6.6**

TABLE III
Linear Phone Classification Accuracy (%) on Librispeech 100 Trained on top of Frozen CPC Features

model	Train Acc	Test Acc
CPC	69.49	69.10
CPC + SE(ReLU)	70.40	69.99
CPC + LorR	**71.58**	**71.18**
CPC + SE(ReLU) + LorR	71.56	71.10

CPC models are pretrained on 100 hours of librispeech data.

TABLE IV
Linear Phone Classification Accuracy (%) on Librispeech 100 Trained on top of Frozen CPC Features

	Test acc
Supervised topline [17]	76.3
Original CPC [14]	65.5
CPC [17]	68.9
CPC + LorR	**71.4**

CPC models are pretrained on 360 hours of librispeech data.

TABLE V
Impact of SE Regularization Weight λ on the ABX Scores

λ	Within	Across
0.2	6.98	9.58
0.4	**6.64**	**9.17**
0.6	6.77	9.31
0.8	6.86	9.75
1.0	6.94	9.64

CPC model was trained on librispeech 100.

portion. All the regularization techniques outperformed the baseline CPC system. Among the different regularization techniques, the LorR worked the best. On average, LorR reduced the ABX error rates by 18% and 16% relative to the baseline in Within and Across conditions, respectively. We also experimented with increasing the number of layers in the autoregressive network, f_{ar}. To observe the impact of model size, we train a LorR regularized CPC system with two LSTM layers. CPC-2L denotes the results with two-layer LSTM as f_{ar}. We observe that with increased model size, the performance improves. In Table I, comparing first row with second, third or fourth row shows the impact of various regularizations and comparing fourth and fifth rows shows the impact of model size.

Table II compares the CPC trained on Librispeech 360 dataset with other existing systems on the Zerospeech 2017 English dataset. We lowered the ABX scores by 15% and 13% relative to the baseline in Within and Across testing conditions compared to the CPC models. As seen in Table II, our regularized models also outperformed the state-of-art feature extraction methods on the ABX task. One key difference is the amount of data used, we train on a larger amount of data, but the data is taken from a different dataset. We also outperformed the supervised topline, which uses posteriorgrams extracted from the supervised HMM-GMM phone recognition system as features. The models trained on 360 hours of data outperformed the models trained on 100 hours of data.

We also trained a linear phone classifier on top of the representations extracted from the CPC models. CPC models were only used as feature extractors and were not finetuned during the phone classifier training. We compared the accuracies of various regularization techniques using the CPC models trained on 100 hours of data. As seen in Table III, similar to the ABX results, linear phone accuracies for the regularized models outperform the baseline CPC model. The LorR regularization performed best among different regularizations. For the regularized CPC systems, we used the same optimal regularization weights discovered on the ABX task. We compared our best regularized CPC system with the best CPC baseline model in Table IV. We outperformed the CPC model and moved towards matching the topline performance.

D. *Hyperparameter Tuning*

For CPC + SE(ReLU) experiments, we varied the regularization weight λ from 0.2 to 1 with a step size of 0.2. Table V shows the ABX scores with different weights. For the SE(ReLU), the optimal weight was 0.4.

For the LorR regularization, we have two hyperparameters, the window size w and the regularization weight α. We varied the regularization weight α from 0.2 to 1 with a step size of 0.2 and trained CPC systems. Table VI shows the ABX scores with different weights. LorR window size two and weight 1.0 performed the best. Please note that all the systems for both SE and LorR, regardless of the choice of weight, outperformed the baseline CPC system.

TABLE VI
IMPACT OF WINDOW SIZE AND LORR LOSS WEIGHT

α	w=2		w=3	
	Within	Across	Within	Across
0.2	6.07	8.65	6.35	9.20
0.4	6.09	8.69	6.15	8.83
0.6	5.98	8.50	6.10	8.51
0.8	6.09	8.57	**5.92**	**8.43**
1.0	**5.86**	**8.29**	5.94	8.71

CPC model was trained on librispeech 100.

E. Mono, Cross and Multilingual Performance

We want to evaluate the performance of the proposed regularization in mono, cross, and multilingual setting. We used Zerospeech 2017 dataset, which contains three languages: English, French, and Mandarin. These three include 45, 24, and 2.5 hours of data, respectively.

We trained the baseline CPC and CPC for each language with LorR regularization. As seen in Tables VII, VIII, and IX, we see consistent improvements in both within-speaker and across-speaker conditions across all the languages on average. For Mandarin (Table VII), we see the slightest improvement in the within-speaker condition. In 1s evaluation, the CPC baseline performed slightly better (11.3) than the CPC + LorR system (11.4). The scores before rounding are 11.34 and 11.36, respectively. In the across-speaker condition, we observe consistent improvements similar to other languages.

For the multilingual setting, we pooled the data from all three languages and trained the baseline CPC and CPC + LorR system on 71.5 (45 + 24 + 2.5) hours of data. The CPC + LorR outperformed the baseline CPC in all three languages in the multilingual setting. We observed that the Multilingual system outperformed the monolingual system for Mandarin (Table VII). We observed little to no performance improvement for French (Table VIII). However, for English (Table IX), the multilingual system performed worse than the monolingual system. We think this is because Mandarin has the smallest amount of data. French and English have a sufficiently large amount of data to train a decent monolingual system.

For the cross-lingual setting, we used the CPC and CPC + LorR system trained on 100 hours portion from the Librispeech dataset. There is a mismatch in the training language and dataset for Mandarin and French, i.e., training is done on Librispeech, whereas testing is done on the Zerospeech 2017 dataset. For English, the only mismatch is in the datasets. As observed in Tables VII, VIII, and IX, the CPC + LorR system outperforms the CPC system across all languages. The CPC/ CPC + LorR trained on Librispeech 100 hours of data outperformed the system trained on individual languages or in a multilingual setting (Tables VII, VIII, and IX). We believe this is because of the amount of data. The total data is less than 100 hours, even in a multilingual setting.

To test our hypothesis, we train three systems on 2.5, 24, and 45 hours subsets from Librispeech 100 hours dataset. For Mandarin (Table VII), the CPC + LorR trained on monolingual data, i.e., Mandarin, outperforms the cross-lingual system trained on Librispeech with the same amounts of data. For French (Table VIII), the monolingual and cross-lingual CPC + LorR systems perform similarly. Increasing the amount of Librispeech data for both languages improves the performance. For English(Table IX), we observe the CPC + LorR system trained on Librispeech 45 hours subset outperforms the system trained on Zerospeech English data (45 hours). Increasing the amount of Librispeech data further improves the performance.

These experiments show the language invariance of the proposed regularization. Across all the training conditions (monolingual, cross-lingual, and multilingual) and various languages (Mandarin, French, and English), CPC + LorR constantly outperformed the baseline CPC.

F. Are SE and LorR Complementary?

We want to analyze whether the regularization techniques, i.e., SE and LorR, are complementary and whether we can use them simultaneously to improve performance further. We train a CPC model which uses both SE(ReLU) and LorR regularization techniques. The total loss of the model is given as

$$\mathcal{L}_{\text{total}} = \mathcal{L}_{\text{CPC}} + 0.5(\alpha\mathcal{L}_{\text{LorR}} + \lambda\mathcal{L}_{\text{SE(ReLU)}}) \tag{8}$$

The weights α and λ for $\mathcal{L}_{\text{LorR}}$ and $\mathcal{L}_{\text{SE(ReLU)}}$ are the best weights from the individual systems, 1 and 0.4 respectively.

We evaluated this system on ABX and linear phone classification. As seen in Tables I and III, it performs better than CPC + SE system but worse than CPC + LorR system. Either the regularization techniques do not have complementary information, or the regularization weights used might not be the optimal choice.

To find the optimal choice for weighing the SE and LorR regularization, we need to train multiple systems. However, that is computationally expensive. We, therefore, simply concatenate the features extracted from different models and train a linear phone classifier. Table X shows phone classification accuracy on various system combinations. The phone classifier trained on top of concatenated features from CPC + SE(ReLU) and CPC + LorR performed the best among the two system combinations. This suggests that complementary information is present in the two features, improving the performance.

G. Clustering Analysis

Unsupervised features are often used for acoustic unit discovery (AUD). In AUD, speech signals are segmented and clustered into discrete phone-like units. We want to test how the representation produced by the regularized CPC cluster compared to baseline CPC. We ran K-means on top of the latent representation extracted from the frozen CPC models trained on Librispeech 100 hours subset. We trained the K-means algorithm on the 100 hours of Librispeech data. We experimented with varying numbers of clusters since the actual number of clusters is unknown.

We use purity and Normalized mutual information (NMI), two commonly used metrics for measuring the clustering quality. As seen from Table XI, the results for CPC + LorR consistently outperformed the baseline CPC. This means the clusters generated

TABLE VII
ABX SCORES ON ZS17 MANDARIN DATASET

	Within				Across			
	1s	10s	120s	avg	1s	10s	120s	avg
CPC (mand)	11.3	11.2	11.3	11.3	13.6	13.3	13.6	13.5
CPC + LorR (mand)	11.4	11.1	11.1	11.2	12.0	12.2	12.3	12.2
CPC + LorR (libri2.5)	12.8	12.9	13.0	12.9	15.2	15.4	15.5	15.4
CPC (eng+fr+mand)	11.0	10.8	11.1	11.0	11.6	11.7	12.3	11.9
CPC + LorR (eng+fr+mand)	9.9	9.6	10.1	9.9	10.1	10.0	10.8	10.3
CPC (libri100)	10.0	9.8	9.8	9.9	9.9	9.6	9.9	9.8
CPC + LorR (libri100)	8.5	8.6	9.1	8.8	8.7	8.5	9.1	8.8

Training data is shown in parentheses.

TABLE VIII
ABX SCORES ON ZS17 FRENCH DATASET

	Within				Across			
	1s	10s	120s	avg	1s	10s	120s	avg
CPC (fr)	12.5	13.1	13.2	12.9	17.8	17.5	17.9	17.7
CPC + LorR (fr)	10.5	10.7	11.0	10.7	15.5	15.0	15.5	15.3
CPC + LorR (libri24)	10.6	10.4	10.6	10.5	15.4	15.1	15.6	15.4
CPC (eng+fr+mand)	12.0	12.1	12.4	12.2	17.3	16.7	17.3	17.1
CPC + LorR (eng+fr+mand)	10.7	10.7	11.1	10.8	15.3	14.6	15.3	15.1
CPC (libri100)	10.7	10.6	10.8	10.7	15.2	14.8	14.9	15.0
CPC + LorR (libri100)	9.6	9.2	10.3	9.7	13.8	13.1	13.7	13.5

Training data is shown in parentheses.

TABLE IX
ABX SCORES ON ZS17 ENGLISH DATASET

	Within				Across			
	1s	10s	120s	avg	1s	10s	120s	avg
CPC (eng)	9.2	9.4	9.6	9.4	13.1	12.9	12.9	13.0
CPC + LorR (eng)	7.5	7.5	7.8	7.6	11.0	10.8	11.0	11.0
CPC + LorR (libri45)	6.2	6.2	6.4	6.3	9.5	9.2	9.4	9.4
CPC (eng+fr+mand)	9.4	9.7	10.0	9.7	13.4	13.3	13.7	13.4
CPC + LorR (eng+fr+mand)	7.9	8.0	8.4	8.1	11.4	11.1	11.4	11.3
CPC (libri100)	7.3	7.1	7.2	7.2	10.3	9.6	9.8	9.9
CPC + LorR (libri100)	5.8	5.6	6.1	5.9	8.4	8.0	8.5	8.3

Training data is shown in parentheses.

TABLE X
LINEAR PHONE CLASSIFICATION RESULTS ON LIBRISPEECH 100 TRAINED ON TOP OF CONCATENATED FEATURES FROM FROZEN CPC MODELS

CPC	CPC + SE	CPC + LorR	Train Acc	Test Acc
✓	-	-	69.49	69.10
✓	✓	-	72.35	71.82
✓	-	✓	73.04	72.54
-	✓	✓	73.39	72.84
✓	✓	✓	74.08	73.54

CPC models are trained on 100 hours of librispeech data.

TABLE XI
PURITY AND NMI (IN PARENTHESIS) ON THE LIBRISPEECH TEST SPLIT FOR A DIFFERENT NUMBER OF CLUSTERS

	25	50	100
CPC	40.5 (38.5)	43.6 (37.7)	46.8 (37.2)
CPC+LorR	**41.2 (40.0)**	**45.2 (40.2)**	**51.2 (40.6)**

by CPC + LorR aligned better with the forced-aligned phone labels than the baseline CPC.

H. Original CPC and Regularization

In the original CPC, a linear transformation W_m is used for predicting $\mathbf{z}_{t+m}$ from $\mathbf{c}_t$. In the modified implementation, a single-layer transformer is used for predicting $\mathbf{z}_{t+m}$. Each transformer layer introduces around 1.3 million new parameters, and there are twelve one-layer transformers in total. The new CPC variant improves the performance significantly but also adds more parameters to the model. We want to see if our regularization works with the old CPC variant.

The regression weights are not used during inference but only during training. The inference model is the same size for the original and the modified variant. The models are trained on Librispeech 100 hours dataset and evaluated on Zerospeech 2017 dataset. As observed in Table XII, our regularized CPC outperformed the original variant. However, the performance was still worse than the modified version of CPC, the original

TABLE XII
PERFORMANCE COMPARISON FOR THE ORIGINAL AND THE MODIFIED IMPLEMENTATION OF THE CPC

	parameters	Within	Across
Original CPC	1.8m	8.6	12.2
Original CPC + LorR	1.8m	7.6	11.0
CPC	17.6m	7.2	9.9
CPC + LorR	17.6m	5.9	8.3

TABLE XIII
SPEAKER CLASSIFICATION PERFORMANCE ON LIBRISPEECH 100

	Acc
CPC	77.0
CPC + LorR	38.8

TABLE XIV
PROBABILITY OF USING UNAUGMENTED DATA

	within	across
0.0	10.2	9.2
0.2	9.9	8.8
0.4	**9.5**	**8.8**
0.6	9.8	9.1
0.8	10.6	10.6
1.0	11.2	12.2

CPC trained on ZS17 mandarin (2.5h) and tested on mandarin test set.

TABLE XV
ABX SCORES ON ZEROSPEECH 2017 CHALLENGE

	English		French		Mandarin	
	W	A	W	A	W	A
CPC (2L) + Aug [30]	6.6	9.3	9.3	14.1	11.2	11.9
CPC (2L) + Aug	6.4	8.8	9.1	13.1	9.8	9.6
CPC (2L) + Aug + LorR	**6.0**	**8.5**	**8.8**	**12.6**	**9.5**	**8.8**

The models are trained on 45, 24, and 2.5 hours of english, french, and mandarin data, respectively. "W" and "A" denotes the within and across speaker scores.

version contained around 10% parameters, which shows the usefulness of the extra parameters.

I. Probing for Speaker Information

As seen from the ABX results in Table I, the performance across-speaker testing conditions are much worse than within-speaker conditions. This implies the features still contain speaker information. Here, we want to analyze how much speaker information is present in the representations. We followed [50] and trained a linear classifier that tries to predict the identity of the speaker from a single frame. We used the last frame from the GRU as the input. It summarizes all the information in the sequence. The speaker classifier was trained for 50 epochs and evaluated on the same train test split used for the linear phone classifier on Librispeech 100.

As seen from Table XIII, the speaker classification performance for CPC + LorR is much lower than the baseline CPC.

TABLE XVI
ABX PERFORMANCE ON ZEROSPEECH 21 ENGLISH DEV SET

	Within		Across	
	clean	other	clean	other
CPC (2L)	6.18	8.46	8.02	13.59
ACPC (2L)	5.37	7.46	7.09	12.60
mACPC (2L)	5.13	-	6.84	-
HCPC (2L)	5.08	-	6.72	-
CPC + LorR (2L)	5.05	7.16	6.48	11.90
CPC + LorR + Aug (2L)	**4.90**	**6.88**	**6.17**	**10.96**

That implies the features extracted from CPC + LorR suppressed speaker information. Even on Mandarin monolingual training Table VII where we get a small improvement (0.1) in the within speaker condition, in the across speaker testing condition, we see good improvement (1.3). We believe this is due to regularization removing some speaker information from the features.

J. Data Augmentation for CPC

Data augmentation techniques greatly improve the performance of the model in supervised scenarios, especially in limited labeled data scenarios. Data augmentation has also become a significant part of SSL systems. Typically, a signal is augmented to generate two views, and the feature extractor is trained to generate the same representation regardless of the augmentation.

The idea is that the underlying class information remains the same regardless of the augmentation. The goal is to learn augmentation invariant representation. By doing so, the representations would capture the underlying class information.

CPC + Aug [30] proposed augmentation of the speech signal in the time domain to improve the CPC performance. At a given point t, the past audio signal, i.e., audio from the beginning till the time t, is fed into the convolutional encoder and then into the autoregressive network to generate the context $\mathbf{c}_t$. The context $\mathbf{c}_t$ is then used to predict future feature frames $\mathbf{z}_{t+k}$.

In the augmented CPC, the past audio signal is corrupted with a noise used for generating the context. This context is then used to predict the feature frames generated by future audio signals corrupted with a different augmentation. This forces the encoder to denoise the data and generate a similar representation regardless of the augmentation. They showed consistent improvements over the baseline CPC. They experimented with augmenting both past and future while training the CPC. Experimentally only augmenting the future while using the unaugmented speech data for the past performed best. The final architecture is shown in Fig. 2.

CPC + Aug experimented extensively with different types of augmentation, their relative order of application, and which portion of the audio signal should be augmented. We followed the experiments in [30] and used pitch shift, noise addition, and reverberation to augment the audio signal. For pitch: we changed the pitch by an integer uniformly sampled between -300 and 300(the change value is measured by 1/100 of a tone). For the additive noise, we used the MUSAN dataset. The additive noise was filtered through a bandpass filter in the [80, 240] Hz range.

TABLE XVII
CROSS LINGUAL TRANSFER OF PRETRAINED FEATURES IN TERMS OF PHONE ERROR RATES

Model	Pretraining	Frozen	du	es	fr	it	ky	ru	sv	tr	tt	zh	Avg
From scratch	-	No	84.7	95.9	95.1	95.0	81.5	97.7	86.1	83.1	72.9	84.3	87.6
Bottleneck	Babel-1070h	Yes	47.9	36.6	48.3	39.0	38.7	45.2	52.6	43.4	42.5	54.3	44.9
Supervised	LS-100h	Yes	42.4	36.4	47.0	40.5	41.0	43.6	47.0	48.5	41.5	56.8	44.5
Orignal CPC	LS-100h	Yes	51.5	44.2	54.5	47.0	44.8	49.0	54.0	54.7	48.9	60.1	50.9
CPC	LS-100h	Yes	44.4	38.7	49.3	42.1	40.7	45.2	48.8	49.7	44.0	55.5	45.8
CPC + LorR	LS-100h	Yes	45.7	36.5	45.9	42.1	40.1	46.5	48.9	48.0	37.4	55.3	44.6
CPC	LS-360h	Yes	**42.5**	38.0	47.1	**40.5**	41.2	**43.7**	**47.5**	47.3	42.0	55.0	44.5
CPC + LorR	LS-360h	Yes	43.3	**35.0**	**43.6**	40.7	**38.9**	44.5	47.7	**46.1**	**35.6**	**54.0**	**42.9**

Training: 1 h data/language from common voice. The languages are: dutch (du), spanish (es), french (fr), italian (it), kyrgyz (ky), russian (ru), sweedish (sv), turkish (tr), tatar (tt) and mandarin (zh).

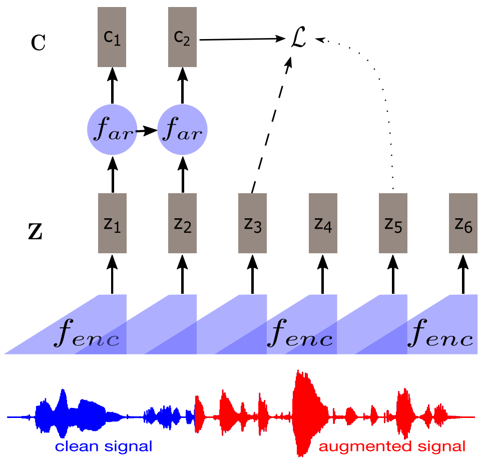

Fig. 2. Overview of the CPC architecture. The solid line represents the reference frame; the dashed line shows the positive, and the dotted line shows the randomly sampled negative example.

For reverb, the room scale was randomly sampled between 0 and 100, and all other parameters were fixed to defaults. We only augment the future audio signal, i.e., both positive and negative samples were generated from the augmented signal. We used WavAugment provided by [30] for the data augmentation experiments.

They also modified the architecture of the CPC: the Ws, instead of being modeled by one layer transformer independently, were all now modeled with a single transformer with $M(= 12)$ classification heads. This improved the training time without significant reductions in performance. For the recurrent context network, a two-layer LSTM is used. For the data augmentation experiments, we follow the same architecture for a fair comparison.

To avoid reading the MUSAN noises in every batch, we preloaded random three seconds noise samples from each noise utterance in MUSAN. In an ideal case, we should load all the utterances from the MUSAN dataset, but that is very memory intensive. While augmenting the speech signal, we randomly sampled a 1.28 seconds signal from the three seconds noise samples and added it to the speech signal. We reload the three seconds noise samples from MUSAN every 100k update steps.

One important question for the augmentation is how much we should augment. We carried out experiments to discover if it is a good idea to augment all the time or augment some times and use unaugmented data the rest of the time. We started by using only unaugmented and increased the probability of using augmented data in steps of 0.2 to using only augmented data. As observed in Table XIV, augmenting all the time can be detrimental to the system's performance. We hypothesize that using clean data allows the model to focus on just feature extraction instead of feature extraction and denoise. Using augmented data all the time still outperforms just using clean data.

The detailed experimental results of our proposed regularization method and the augmented CPC baseline are in Table XV. As evident from the table, regularization and augmentation can be complementary, and we see consistent improvements over the augmented CPC baseline across all three languages.

Our implementation of the augmented CPC performed better than the reported number in the augmented CPC paper. We believe this is due to the following differences: our strategy for additive noise is different. We effectively sample from a smaller pool of additive noises. There might also be differences in the number of training epochs for the reported numbers for the augmented CPC system vs. our implementation.

K. Comparison With Other CPC Variants

The 2021 version of the Zerospeech challenge [35] used Librispeech for training and evaluation for the ABX task. We compare our system with the small budget CPC baseline, which was trained on Librispeech 100 hours subset. Another method we compared is ACPC [25], which improves over the CPC system. All the systems use two-layer LSTMs in the f_{ar}.

Multilevel systems such as mACPC [26] and HCPC [27] outperform CPC and ACPC but also introduce more parameters and complexity to the model. For example, the architecture for the lower CPC in HCPC is the same as our models, but it also has additional layers for the higher level CPC, the boundary detector, etc. It also requires finetuning the reinforcement learning-based boundary detector and the quantization module.

Our method is more straightforward and does not introduce more parameters to CPC. As seen in Table XVI, our systems outperforms the existing single-level systems, such as CPC [14], [17] and ACPC [25], and multilevel systems, such as mACPC [26] and HCPC [27]. Adding augmentation further improves the system performance, especially in the across-speaker setting.

L. Cross Lingual Phone Recognition

CPC models are commonly used as feature extractors for downstream tasks such as automatic speech recognition. As evident from the linear phone classification experiments, CPC features capture the phone information quite well. We want to analyze how well these features work in different languages. We follow the experiments in [17] and consider the task of phone classification on different languages from the common voice dataset. Phone recognizers are trained with CTC objective.

To show the advantage of using pretrained feature extractors in a low-resource setting, we used the model trained only on the 1-hour target dataset. Table XVII shows the phone error rates on the CommonVoice dataset. The models trained from scratch perform poorly. The models trained on top of the pretrained feature extractor work significantly better across all languages. In both 100 and 360 hours training settings, our regularized CPC models outperformed the baseline CPC models. Our regularized CPC systems trained with just 100 hours of unlabeled data almost matched the performance of CPC trained with 360 hours of data.

Unsupervised feature extractors typically require more data to match the performance of supervised pre-trained feature extractors [15], [17]. While CPC needed 360 hours of unlabeled data to match the quality of supervised pretraining, our regularized CPC matched the performance with just 100 hours of data. With 360 hours of data, our models outperformed the supervised pretraining, which is significant as it is easier to collect unlabeled data than labeled data.

V. Conclusion and Future Work

SSL methods such as CPC have become the front end of speech technologies. ASR systems trained on top of SSL features require much less labeled data to achieve the same performance as models trained from raw speech data. We can further lower the labeled data requirements by improving the feature extraction methods. In this work, we propose regularization techniques that impose slowness constraints on the features learned by a CPC model. We compared our proposed methods with the CPC baseline on ABX, linear phone classification, clustering, and phone recognition tasks. Our models outperformed the baseline CPC models on all the tasks in monolingual, cross-lingual, and multilingual settings. Left-or-Right regularization performs the best among the proposed regularizations. We also compared our methods with recent variants of CPC like ACPC and mACPC and showed that our methods outperform them all.

In the future, we would like to apply these techniques on a larger scale, i.e., training on all of Librispeech. We would also like to use the proposed regularization techniques for other feature extraction techniques, such as Wav2Vec2.0. Multistage methods such as SCPC and mACPC apply segmental constraints via a second-level CPC. We would also like to see if our regularization techniques can improve the multilevel systems discrimination performance. Another interesting direction would be to apply the proposed constraints in a semi-supervised or supervised scenario.

References

[1] G. Hinton et al., "Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups," *IEEE Signal Process. Mag.*, vol. 29, no. 6, pp. 82–97, Nov. 2012.

[2] N. Dehak, P. J. Kenny, R. Dehak, P. Dumouchel, and P. Ouellet, "Front-end factor analysis for speaker verification," *IEEE Trans. Audio, Speech, Lang. Process.*, vol. 19, no. 4, pp. 788–798, May 2011.

[3] O.-W. Kwon, K. Chan, J. Hao, and T.-W. Lee, "Emotion recognition by speech signals," in *Proc. 8th Eur. Conf. Speech Commun. Technol.*, 2003, pp. 125–128.

[4] A. Jansen and B. Van Durme, "Efficient spoken term discovery using randomized algorithms," in *Proc. IEEE Workshop Autom. Speech Recognit. Understanding*, 2011, pp. 401–406.

[5] L. Badino, C. Canevari, L. Fadiga, and G. Metta, "An auto-encoder based approach to unsupervised learning of subword units," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process.*, 2014, pp. 7634–7638.

[6] B. Varadarajan, S. Khudanpur, and E. Dupoux, "Unsupervised learning of acoustic sub-word units," in *Proc. 46th Annu. Meeting Assoc. Comput. Linguistics Hum. Lang. Technol., Short Papers*, 2008, pp. 165–168.

[7] M. Huijbregts, M. McLaren, and D. Van Leeuwen, "Unsupervised acoustic sub-word unit detection for query-by-example spoken term detection," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process.*, 2011, pp. 4436–4439.

[8] C.-Y. Lee and J. Glass, "A nonparametric Bayesian approach to acoustic model discovery," in *Proc. 50th Annu. Meeting Assoc. Comput. Linguistics, Long Papers*, 2012, vol. 1, pp. 40–49.

[9] M.-H. Siu, H. Gish, A. Chan, W. Belfield, and S. Lowe, "Unsupervised training of an HMM-based self-organizing unit recognizer with applications to topic classification and keyword discovery," *Comput. Speech Lang.*, vol. 28, no. 1, pp. 210–223, 2014.

[10] H. Kamper, A. Jansen, and S. Goldwater, "A segmental framework for fully-unsupervised large-vocabulary speech recognition," *Comput. Speech Lang.*, vol. 46, pp. 154–174, 2017.

[11] S. Bhati, S. Nayak, and K. S. R. Murty, "Unsupervised speech signal to symbol transformation for zero resource speech applications," in *Proc. Annu. Conf. Int. Speech Commun. Assoc.*, 2017, pp. 2133–2137.

[12] H. Kamper, K. Livescu, and S. Goldwater, "An embedded segmental k-means model for unsupervised segmentation and clustering of speech," in *2017 IEEE Autom. Speech Recognit. Understanding Workshop*, 2017, pp. 719–726.

[13] S. Bhati, H. Kamper, and K. S. R. Murty, "Phoneme based embedded segmental k-means for unsupervised term discovery," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process.*, 2018, pp. 5169–5173.

[14] A. V. d. Oord, Y. Li, and O. Vinyals, "Representation learning with contrastive predictive coding," in *Proc. Adv. Neural Inf. Process. Syst.*, 2018, pp. 1–10.

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

[16] A. Baevski, H. Zhou, A. Mohamed, and M. Auli, "wav2vec 2.0: A framework for self-supervised learning of speech representations," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 33, 2020, pp. 12449–12460.

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

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

[19] Y. Liu et al., "RoBERTa: A robustly optimized bert pretraining approach," 2019, *arXiv:1907.11692*.

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

[21] M. Caron et al., "Emerging properties in self-supervised vision transformers," in *Proc. IEEE/CVF Int. Conf. Comput. Vis.*, 2021, pp. 9650–9660.

[22] M. Hallap, E. Dupoux, and E. Dunbar, "Evaluating context-invariance in unsupervised speech representations," *Interspeech*, pp. 2973–2977, 2023.

[23] 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.

[24] S. Bhati, J. Villalba, P. Żelasko, L. Moro-Velázquez, and N. Dehak, "Segmental contrastive predictive coding for unsupervised word segmentation," in *Proc. Annu. Conf. Int. Speech Commun. Assoc.*, 2021, pp. 366–370.

[25] J. Chorowski et al., “Aligned contrastive predictive coding,” *Interspeech*, pp. 976–980, ISCA, 2021.

[26] S. Cuervo et al., “Contrastive prediction strategies for unsupervised segmentation and categorization of phonemes and words,” in *Proc. ICASSP 2022-2022 IEEE Int. Conf. Acoust., Speech, Signal Process.*, 2022, pp. 3189–3193.

[27] S. Cuervo, A. Lancucki, R. Marxer, P. Rychlikowski, and J. K. Chorowski, “Variable-rate hierarchical CPC leads to acoustic unit discovery in speech,” in *Proc. Int. Conf. Adv. Neural Inf. Process. Syst.*, 2022, vol. 35, pp. 34995–35006.

[28] S. Bhati, J. Villalba, P. Żelasko, and N. Dehak, “Self-expressing autoencoders for unsupervised spoken term discovery,” *Interspeech*, pp. 4876–4880, 2020.

[29] D. S. Park et al., “Specaugment: A simple data augmentation method for automatic speech recognition,” *Interspeech*, pp. 2613–2617, 2019.

[30] E. Kharitonov et al., “Data augmenting contrastive learning of speech representations in the time domain,” in *Proc. IEEE Spoken Lang. Technol. Workshop*, 2021, pp. 215–222.

[31] M. Versteegh et al., “The zero resource speech challenge 2015,” in *Proc. Annu. Conf. Int. Speech Commun. Assoc.*, 2015, pp. 3169–3173.

[32] M. Versteegh, X. Anguera, A. Jansen, and E. Dupoux, “The zero resource speech challenge 2015: Proposed approaches and results,” *Procedia Comput. Sci.*, vol. 81, pp. 67–72, 2016.

[33] E. Dunbar et al., “The zero resource speech challenge 2017,” in *Proc. Autom. Speech Recognit. Understanding Workshop*, 2017, pp. 323–330.

[34] E. Dunbar et al., “The zero resource speech challenge 2019: TTS without T,” in *Proc. Interspeech 2019-20th Annu. Conf. Int. Speech Commun. Assoc.*, 2019, pp. 1–10.

[35] E. Dunbar et al., “The zero resource speech challenge 2021: Spoken language modelling,” in *Proc. NeuRIPS Workshop Self-Supervised Learn. Speech Audio Process.*, 2020, pp. 1–9.

[36] H. Kamper, “Truly unsupervised acoustic word embeddings using weak top-down constraints in encoder-decoder models,” in *Proc. ICASSP 2019-2019 IEEE Int. Conf. Acoust., Speech, Signal Process.*, 2019, pp. 6535–3539.

[37] D. Renshaw, H. Kamper, A. Jansen, and S. Goldwater, “A comparison of neural network methods for unsupervised representation learning on the zero resource speech challenge,” in *Proc. Annu. Conf. Int. Speech Commun. Assoc.*, 2015, pp. 3199–3203.

[38] D. P. Kingma and M. Welling, “Auto-encoding variational bayes,” 2013, *arXiv:1312.6114*.

[39] A. Van Den Oord et al., “Neural discrete representation learning,” in *Proc. Int. Conf. Adv. Neural Inf. Process. Syst.*, 2017, vol. 30, pp. 6309–6318.

[40] F. Kreuk, J. Keshet, and Y. Adi, “Self-supervised contrastive learning for unsupervised phoneme segmentation,” in *Proc. Interspeech*, 2020, pp. 3700–3704.

[41] B. van Niekerk, L. Nortje, and H. Kamper, “Vector-quantized neural networks for acoustic unit discovery in the zerospeech 2020 challenge,” in *Proc. Interspeech*, 2020, pp. 4836–4840.

[42] L. Wiskott and T. J. Sejnowski, “Slow feature analysis: Unsupervised learning of invariances,” *Neural Comput.*, vol. 14, no. 4, pp. 715–770, 2002.

[43] J. Chorowski et al., “Unsupervised neural segmentation and clustering for unit discovery in sequential data,” in *Proc. NeurIPS 2019 Workshop-Percep. Generative Reasoning-Struct., Causality, Probability*, 2019, pp. 1–7.

[44] H. Kamper and B. van Niekerk, “Towards unsupervised phone and word segmentation using self-supervised vector-quantized neural networks,” in *Proc. Interspeech*, 2021, pp. 1539–1543..

[45] H. Kamper, “Word segmentation on discovered phone units with dynamic programming and self-supervised scoring,” *IEEE/ACM Trans. Audio, Speech, Lang. Process.*, vol. 31, pp. 684–694, 2023.

[46] S. Bhati, J. Villalba, P. Żelasko, L. Moro-Velazquez, and N. Dehak, “Unsupervised speech segmentation and variable rate representation learning using segmental contrastive predictive coding,” *IEEE/ACM Trans. Audio, Speech, Lang. Process.*, vol. 30, pp. 2002–2014, 2022.

[47] T. Schatz, V. Peddinti, F. Bach, A. Jansen, H. Hermansky, and E. Dupoux, “Evaluating speech features with the minimal-pair ABX task: Analysis of the classical MFC/PLP pipeline,” in *Proc. 14th Annu. Conf. Int. Speech Commun. Assoc.*, 2013, pp. 1–5.

[48] A. Vaswani et al., “Attention is all you need,” in *Proc. Int. Conf. Adv. Neural Inf. Process. Syst.*, 2017, vol. 30, pp. 6000–6010.

[49] M. Heck, S. Sakti, and S. Nakamura, “Feature optimized DPGMM clustering for unsupervised subword modeling: A contribution to zerospeech 2017,” in *Proc. IEEE Autom. Speech Recognit. Understanding Workshop*, 2017, pp. 740–746.

[50] B. van Niekerk, L. Nortje, M. Baas, and H. Kamper, “Analyzing speaker information in self-supervised models to improve zero-resource speech processing,” *Interspeech*, 2021, pp. 1554–1558.