The Government of the Russian Federation
Federal State Autonomous Institution for Higher Education
National Research University Higher School of Economics
St. Petersburg Branch
St. Petersburg School of Physics, Mathematics and Computer Science

Alena Pestova

Speech Representation Learning with Contrastive Predictive Coding

Master's Thesis
Area of studies 01.04.02 «Applied Mathematics and Informatics»
Master Program "Machine Learning and Data Analysis"

Scientific supervisor:
professor Alexander Omelchenko

Consultant:
Alexander Samarin

Reviewer:
Anna Senyaeva

Saint-Petersburg
2023

Contents

Introduction

Self-supervised models have become increasingly popular in recent years due to their ability to learn from unlabeled data, making them more efficient and cost-effective than traditional supervised learning methods. These models have been successfully applied to various fields, including computer vision and natural language processing.

Self-supervised models are used in the field of speech processing as well. Examples of such models are Wav2vec2 [1] and HuBERT [2]. However, the main trend in the development of self-supervised models is an increase in the number of parameters. Thus, the big size of such models can be a problem for their use in real product tasks with clear restrictions on the size, for example, in different speech processing tasks in mobile phones.

In this work, I experiment with such a method for obtaining vector speech representations as Contrastive Predictive Coding. The original model architecture proposed by [3] does not include the use of heavy modules. However, from existing research, it is not clear how generalizable are representations from CPC and whether this method can be used for very different speech-processing tasks. Similar models are mostly evaluated in terms of speaker identification and phoneme discriminability on different datasets with varying metrics.

So, **the goal of my work** is to explore the applicability of Contrastive Predictive Coding models for learning generalizable speech representations and evaluate their competitiveness with large language models. In order to do that, the following **tasks** were set:

- To reproduce the CPC model from the original article.
- To experiment with the architecture of the original model, with increasing the number of parameters and training dataset.
- To compare the CPC models with each other and with baselines on a wide range of downstream tasks.

1 Literature Review

1.1 Motivation for Self-Supervised Learning

Classic deep learning models are trained in a supervised way - they are trained to map some input features of observations to their labels. However, the need for labeled data for model training becomes a bottleneck in supervised learning. Manually annotated data is expensive and time-consuming to produce. This can be an obstacle to scaling deep neural networks as bigger models need more data to train. What is more, annotated data may be available in small quantities or not available at all in narrow areas, rare languages, etc.

This is where the idea of self-supervised learning comes in. Self-supervised learning is quite similar to usual supervised learning - a model again learns how to map input features and data labels, but the difference is that the labels are derived from the data itself. More often it is done by using some part of the input as a label. For example, the model BERT [4] from Natural Language Processing (NLP) is trained to predict randomly masked words in the input sequence. Another popular task is language modeling, when some model (like the transformer-based GPT model [5]) learns to predict (generate) the next word for the current sequence. These models are also trained in a self-supervised manner. The advantage of this approach is that there is a huge amount of raw unlabeled data that can be derived from the Internet, and it can be used for pre-training big neural networks.

However, such models are useful not only in the context of their training tasks like language modeling or gap filling. Their true advantage is the opportunity for transfer learning. The general idea is that the model can be trained on a big amount of unlabeled data in a self-supervised manner and then used in some other downstream tasks (either with frozen parameters or with finetuned). For example, the BERT model is widely used as a feature extractor in very different NLP tasks like Named Entity Recognition, Question Answering, Natural Language Understanding, etc. [1] This is also

[1] `https://paperswithcode.com/paper/bert-pre-training-of-deep-bidirectional`

an important feature of self-supervised learning - the model can be then used not only for one specific task but we can use its representations for solving very different problems.

1.2 Contrastive Representation Learning

Contrastive Representation Learning is one of the popular methods used for training the models in a self-supervised manner.

There are two main groups of methods to learn data representations - generative and discriminative modeling [6]. Generative approaches refer to modeling data distribution $p(x)$. For example, in the case of speech processing, it could be generating the next timestamps of audio. Discriminative methods are based on learning distribution $p(y|x)$ through inferring input to latent variable v and using it for prediction (x is the input data and y is some label).

The difference between contrastive learning methods and these approaches is that it is based on a comparison of data samples. This comparison is done between some positive (similar) examples and negative (not similar) ones. The aim of contrastive learning is quite simple: we want representations of similar data to be close, while for different samples they should be further away in vector space. Instead of using one sample at a time for learning data distribution, several samples and their comparison are used.

More formally, contrastive learning needs query q, positive k^+, and negative k^- keys. Thus, (q, k^+) would be positive examples that we want to be similar (close in vector space), while (q, k^-) would be a negative pair which parts need to be distinct (far away from each other in a vector space). The way how these pairs are constructed is one of the most important parts of contrastive learning as it defines the model's objective. For example, the task could be to predict a center word (k^+) by the surrounding words (q) with other randomly sampled words taken as k^- (as it is done in the model *word2vec* [7]). In the case of Computer Vision, the objective could be to identify that two augmented images (q and k^+) represent the same image in comparison to some other randomly sampled one (k^-) (like in *SimCLR* [8]).

As for the speech processing, the task could be to predict the representation of a future signal (k^+) by the contextualized embedding of the current step (q) taking random signals as k^-. This is how it is done in the CPC model discussed in the work as well as the other models that follow CPS's idea (*wav2vec* [9] and *VQ-wav2vec* [10]).

It is also worth mentioning, what type of losses and objectives are used in contrastive learning. In the paper [6], the authors define 3 main groups of contrastive loss: energy-based, noise contrastive estimation-based, and mutual information-based.

Training energy-based models (EBM) are directed at associating low distance with positive examples and high distance with negative ones. It is quite similar to what NCE-based loss does, however, lowering the distance between desired pairs does not make negative examples further away. That's why usually such models need separate comparisons with negative examples. The most popular examples of EBM are pair and triplet loss.

The pair loss [11, 12] directly minimizes the Euclidian distance $d(q, k)$ between positive pairs and maximizes it between negative pairs by forcing it to be more than some predefined margin $m > 0$. The formula is:

$$L = \begin{cases} d(q,k)^2, \text{if k - positive key} \\ \max(0, m - d(q,k)^2), \text{if k - negative key} \end{cases} \tag{1}$$

The triplet loss is quite similar to the pair loss. The difference is that while the pair loss optimizes the distances between positive and negative examples separately and needs the distance for the negative key to be just bigger than the margin, the triplet loss optimizes the relative distance between the query, positive key, and negative key together:

$$L = max(0, m + d(q,k^+)^2 - d(q,k^-)^2) \tag{2}$$

Although popular in contrastive learning, the pair and triplet loss often converge slowly due to a lack of interactions between samples [6].

Noise contrastive estimation loss will be discussed separately in the next section as it is used in CPC training.

Mutual information-based (MI-based) losses are based on the minimization of mutual information using different MI estimators. What is more, the authors of the paper about CPC [3] proved that minimizing their NCE-based InfoNCE loss minimizes mutual information between the signal and its encoding as well.

1.3 Noise Contrastive Estimation

Loss for training the CPC model that will be discussed later is strongly connected to Noise Contrastive Estimation (NCE) [13]. So, its idea will be described shortly in this subsection.

We can look at the contrastive learning problem like the softmax classification problem. If we remember the classic supervised softmax classification objective, then the formula for the probability that some input x was correctly classified as i class from the set of n classes with z being prediction logits will be:

$$p(i|x) = \frac{\exp(z_i)}{\sum_{j=1}^{n} \exp(z_j)} \tag{3}$$

If we present this softmax function as the non-parametric version which gives us the probability of correctly identifying the positive example k^+ for a given query q from a set K of all possible keys k, then we will have:

$$p(k^+|q) = \frac{\exp(q^T k^+)}{\sum_{k \in K} \exp(q^T k)} \tag{4}$$

Thus, the objective would be minimizing the following loss:

$$L = -\log p(k^+|q) \tag{5}$$

We could use this softmax formula for estimation, however, there is a problem that we need to sum over all possible keys in the denominator to obtain the probability for a given query and positive keys which is a too computationally expensive operation.

NCE gives us a solution for this problem - instead of using all the possible keys here, we can replace our objective with the binary classification task of

discriminating positive data samples and noise (negative) samples. Thus, there will be no need in summing up the values for all the possible examples, instead, only sampled negative keys will be used in the denominator.

So, the NCE uses the probability of a positive sample to be correctly identified as follows:

$$p(i = positive|q, k) = \frac{\exp(q^T k^+)}{\exp(q^T k^+) + \sum_{k \in K_{neg}} \exp(q^T k^-)} \tag{6}$$

where k^- is the negative example, K_{neg} is the set of negative examples we sampled.

Then, this probability is used in the usual binary cross-entropy (BCE) loss which is optimized.

There are also modifications of NCE loss, for example, [14] uses the softmax function for the evaluation of probability by shifting the task to identifying and ranking the positive example with the highest similarity to the query. The authors [8] use temperature in probability calculations for controlling the sensitivity of the similarity function used.

1.4 Contrastive Predictive Coding

Contrastive Predictive Coding (CPC) is the self-supervised method for learning data vector representations proposed by [3]. The authors combine the ideas of predictive coding with contrastive learning.

The method uses a contrastive learning approach to learn - its loss is based on Noise Contrastive Estimation.

As for predictive coding, this is a popular approach to training self-supervised models. Its idea is based on predicting some parts of the data, for example, future samples or masked data. CPC utilizes the idea of predictive coding as predicting encoded representations of future timestamps but not the signals themselves.

Unlike in generative models, where each detail is reconstructed, here the task is to obtain good representations not generated data. This is quite an important and useful idea as predicting the input itself can be compu-

tationally intense, but if our goal is not to build a model generating new data, doing this seems to be not optimal. So, we can still use a prediction of future timestamps as a learning objective but instead of predicting the data itself, we can predict its vector representation of lower dimensionality. This is exactly what the CPC model does - it learns to predict encoded representations of future timestamps.

So, the idea is to maximize the mutual information between the target signal x (future timestamp) and current context embedding c (present timestamp):

$$I(x, c) = \sum_{x,c} p(x, c) \log \frac{p(x|c)}{p(x)} \tag{7}$$

More formally, the architecture of CPC proposed by [3] is the following. The signal x is encoded in lower-dimensional latent space with some encoder $z = g_{enc}(x)$. Then, it goes to an autoregressive model g_{ar} and we obtain summarized context representations $c_t = g_{ar}(z_t), z < t$ (Figure 1).

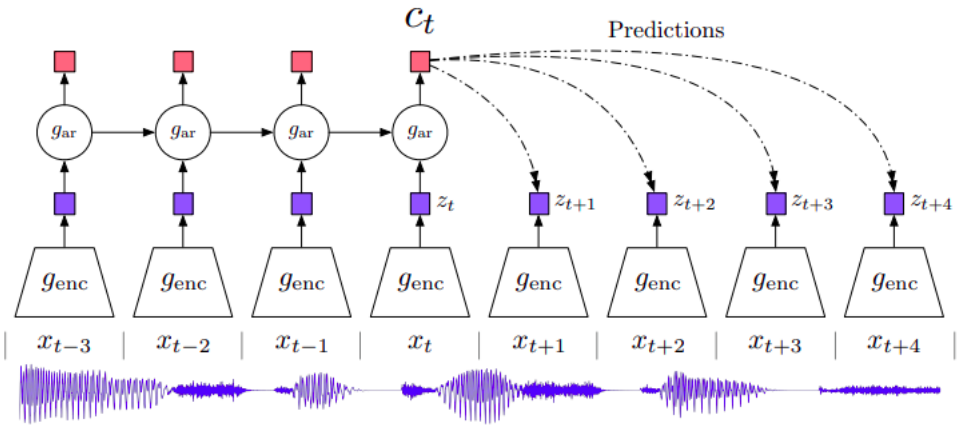

Figure 1: Visualization of predictive coding [3].

Then, k future timestamps are predicted from c_t with linear transformations: $z_{pred\_t+k}(c_t) = W_k c_t$ for each k.

So, instead of predicting x_{t+k} directly and modelling the distributions $p(x_{t+k})$ and $p(x_{t+k}|c)$ (which would correspond to a generative model), the following density ratio is modeled:

$$f_k(x_{t+k}, c_t) \propto \frac{p(x_{t+k}|c_t)}{p(x_{t+k})} \tag{8}$$

It preserves the mutual information between x_{t+k} and c_t as will be shown later.

This expression cannot be computed precisely, however, it can be approximated with a log-bilinear model as proposed by the authors:

$$f_k(x_{t+k}, c_t) = \exp(z_{t+k}^T W_k c_t)$$

, where

$$z_{t+k} = g_{enc}(x_{t+k})$$

As was mentioned above, the authors use simple linear transformations as W_k here. However, this is not necessary and other non-linear or recurrent networks can be used here. For example, in the paper [15] 1-layer Transformer is used instead.

1.5 InfoNCE loss

As was mentioned earlier, the aim of contrastive learning is to make vectors for similar signals closer while making different signals further. CPC model makes similar signals closer by maximizing $f_k(x_{t+k}, c_t)$. In simple words, it maximizes the mutual information between representations of the prediction of the timestamp encoding and the real encoding of this signal. This encoded timestamp will be referred to as a *positive example* in the future.

Contrary to positive examples, another goal of the model is to minimize the MI between some encoded randomly sampled timestamps and encodings predicted based on different context vectors: $f_k(x_n, c_t)$, where $c_t \neq g_{ar}(g_{enc}(x_n))$. These will be referred to as *negative examples* in the future.

These positive and negative examples are necessary because we do not model the distributions $p(x_{t+k})$ and $p(x_{t+k}|c_t)$ directly. Instead, we use these examples as samples from the distributions. In this case, the positive example represents a sample from $p(x_{t+k}|c_t)$, while the negative examples

are from $p(x_{t+k})$ distribution.

The CPC model is trained by minimizing loss that the authors call InfoNCE loss which is based on Noise Contrastive Estimation (NCE). For a given $X = x_1, \ldots, x_n$ consisting of one positive example x_t and $N-1$ negative examples sampled from the batch, this contrastive loss minimizes the dot product between the predicted embedding $z_{pred_{t+k}}(c_t)$ from the future and the real encoded embedding of this part z_{t+k}, while maximizing the dot product between the same prediction $z_{pred_{t+k}}(c_t)$ and negative examples:

$$L_N = - \mathop{\mathbb{E}}_{x \in X} \log \frac{f_k(x_{t+k}, c_t)}{\sum_{x \in X} f_k(x_n, c_t)} = - \mathop{\mathbb{E}}_{x \in X} \log \frac{\exp(z_{t+k}^T z_{pred_{t+k}})}{\sum_{z_n = g_{enc}(x_n), x \in X} \exp(z_n^T z_{pred_{t+k}})} \tag{9}$$

So, in this case, $\frac{f_k(x_{t+k}, c_t)}{\sum_{x \in X} f_k(x_n, c_t)}$ is considered as the probability of correctly predicting the positive sample.

Maximizing this loss will lead to the estimation of density ratio from Equation 8. This can be proven as follows:

loss in Equation 9 can be seen as the categorical cross-entropy loss with prediction $\frac{f_k}{\sum_X f_k}$ for the positive class. Let $p(i = \text{positive}|X, c_t)$ be the optimal probability of this loss where $i = \text{positive}$ denotes the identifier function that a sample is positive. Then, we can write a probability that some sample x_i is a positive example (came from $p(x_{t+k}|c_t)$ distribution) rather than negative one (drawn from $p(x_{t+k})$):

$$p(i = \text{positive}|X, c_t) = \frac{p(x_i|c_t) \prod_{l \neq i} p(x_l)}{\sum_{j=1}^{N} p(x_j|c_t) \prod_{l \neq j} p(x_l)} = \frac{\frac{p(x_i|c_t)}{p(x_i)}}{\sum_{j=1}^{N} \frac{p(x_j|c_t)}{p(x_j)}} \tag{10}$$

In the nominator here we see the probabilities for x being from $p(x_{t+k}|c_t)$ and all other examples from $p(x_{t+k})$ (these probabilities are multiplied). However, we also need to divide this probability by the sum of all other similar calculated probabilities for all other sampled examples.

As it can be seen from the Equation 10, the optimal value for $f_k(x_{t+k}, c_t)$ is proportional to $\frac{p(x_{t+k}|c_t)}{p(x_{t+k})}$. So, optimization of the InfoNCE Loss will lead

to maximization of $\frac{p(x_{t+k}|c_t)}{p(x_{t+k})}$ and, therefore, of the density rate $f_k(x_{t+k}, c_t)$.

InfoNCE loss and Mutual Information. We show that optimization of InfoNCE loss leads to the maximization of $f_k(x_{t+k}, c_t)$. The only thing left without proof for this time is the statement that the density ratio $f_k(x_{t+k}, c_t)$ preserves MI between x_{t+k} and c_t - $I(x_{t+k}, c_t)$. In other words, we need to prove that loss optimization lead to the maximization of $I(x_{t+k}, c_t)$.

We can represent the optimal value of the loss from Equation 9 by replacing $f_k(x_{t+k}, c_t)$ with the $\frac{p(x_{t+k}|c_t)}{p(x_{t+k})}$ and separating X into negative and positive examples in the denominator:

$$L_N^{opt} = - \mathop{\mathbb{E}}_{x \in X} \log \left[\frac{\frac{p(x_{t+k}|c_t)}{p(x_{t+k})}}{\sum_{x \in X} \frac{p(x_n|c_t)}{p(x_n)}} \right] =$$

$$= - \mathop{\mathbb{E}}_{x \in X} \log \left[\frac{\frac{p(x_{t+k}|c_t)}{p(x_{t+k})}}{\frac{p(x_{t+k}|c_t)}{p(x_{t+k})} + \sum_{x \in X \setminus x_{t+k}} \frac{p(x_n|c_t)}{p(x_n)}} \right]$$

Substituting a minus into the expression and deriving $\frac{p(x_{t+k})}{p(x_{t+k}|c_t)}$ from the sum, we get:

$$= \mathop{\mathbb{E}}_{x \in X} \log \left[1 + \frac{p(x_{t+k})}{p(x_{t+k}|c_t)} \sum_{x \in X \setminus x_{t+k}} \frac{p(x_n|c_t)}{p(x_n)} \right] =$$

$$\approx \mathop{\mathbb{E}}_{x \in X} \log \left[1 + \frac{p(x_{t+k})}{p(x_{t+k}|c_t)} (N-1) \mathop{\mathbb{E}}_{x_n} \frac{p(x_n|c_t)}{p(x_n)} \right]$$

And we can reduce the following part of the equation:

$$\mathop{\mathbb{E}}_{x} \frac{p(x|c)}{p(x)} dx = \int \frac{p(x|c)}{p(x)} p(x) dx = \frac{1}{p(c)} \int \frac{p(x,c)}{p(x)} p(x) dx =$$

$$= \frac{1}{p(c)} \int p(x,c) dx = \frac{1}{p(c)} p(c) = 1$$

So, returning to the previous equation, we have:

$$\mathop{\mathbb{E}}_{x \in X} \log \left[1 + \frac{p(x_{t+k})}{p(x_{t+k}|c_t)} (N-1) \mathop{\mathbb{E}}_{x_n} \frac{p(x_n|c_t)}{p(x_n)} \right] =$$

$$= \underset{x \in X}{\mathbb{E}} \log \left[1 + \frac{p(x_{t+k})}{p(x_{t+k}|c_t)}(N-1) \right] =$$

Since $p(x_{t+k}) \leq p(x_{t+k}|c_t)$,

$$\underset{x \in X}{\mathbb{E}} \log \left[1 + \frac{p(x_{t+k})}{p(x_{t+k}|c_t)}(N-1) \right] >= \underset{x \in X}{\mathbb{E}} \log \left[\frac{p(x_{t+k})}{p(x_{t+k}|c_t)} N \right] =$$

$$= \underset{x \in X}{\mathbb{E}} \log \left[\frac{p(x_{t+k})}{p(x_{t+k}|c_t)} \right] + \underset{x \in X}{\mathbb{E}} \log N =$$

$$= - \underset{x \in X}{\mathbb{E}} \log \left[\frac{p(x_{t+k}|c_t)}{p(x_{t+k})} \right] + \underset{x \in X}{\mathbb{E}} \log N =$$

$$= -I(x_{t+k}, c_t) + \log N$$

To sum up,

$$I(x_{t+k}, c_t) \geq \log N - L_N^{opt} \tag{11}$$

where $N-1$ is the number of negative samples.

Although previous discussions of InfoNCE loss optimization are not dependent on the number of negative examples $N-1$, we can see that this number influences the lower bound of MI. What is more, it can be seen from Equation 11 that when the loss is minimized, the lower bound for MI between x_{t+k} and c_t also increases.

1.6 Extracting audio features from CPC

The CPC model is useful only in terms of using its trained representations in some other downstream tasks. So, it should be mentioned how exactly these vectors can be extracted.

The authors [3] propose to use either z or c as embeddings. Encoder embeddings $z = g_{enc}(x)$ contain representations for each 10 ms of the given audio. However, each embedding represents only its window and does not capture information from the previous steps.

The output from RNN $c = g_{ar}(z)$ contains embeddings for each 10 ms of

the audio as well, but each $c_t = g_{ar}(z_{<t})$ also have information for previous steps which can be beneficial when the context is important.

In the case when it is necessary to obtain one representation for the whole sequence, it is proposed to pool audio embedding from z or c over all timestamps. This is relevant for such tasks as speaker identification or other sequence classification problems.

The authors used c representations for classification experiments. However, they do not experiment with using z representations instead, so it will be done in this work that will be discussed later.

1.7 Data Augmentation for CPC

There was no data augmentation in the experiments in the original paper [3]. However, data augmentation is a very important part of training neural networks as it can increase the size of the dataset which may improve the performance of the model and reduce overfitting.

The authors [16] use different audio augmentation techniques and their combinations when training the CPC models, and show that it improves the performance of CPC in terms of phoneme discriminability. In more detail, they experiment with combinations of the following audio transformations: pitch modification (*pitch*), additive noise (*add*), reverberation (*reverb*), band reject filtering (*bandrej*), and time masking (*tdrop*).

However, what is more interesting, the authors discuss an approach to applying different augmentation functions to different parts of the data. In more detail, we can separate the *past* and *future* parts of the signal on each step (Figure 2). *Past* part refers to the audio timestamps $x_{<t}$ that are used then for predicting future timestamps. *Future* part refers to the target audio timestamps which representations a model tries to predict in this step. These are positive and negative examples that are then used in contrastive loss. So, we can distinguish 3 groups of audio timestamps: *past*, *future-positive*, and *future-negative*. *past* and *future-positive* frames are from the same sequence, while *future-negative* came from some other timestamp/sequence/speaker. The idea here is that we can now apply different

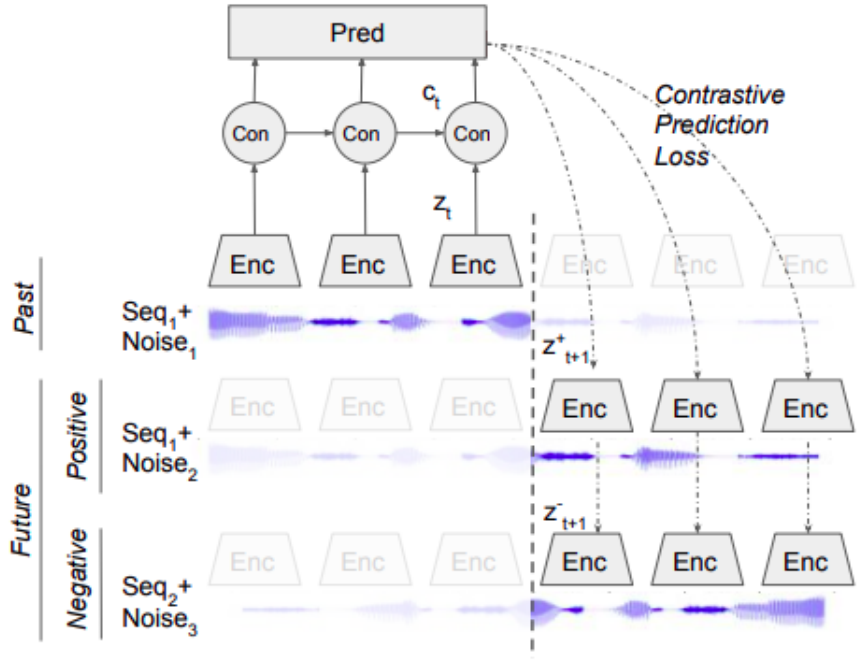

Figure 2: Vizualization of CPC [16].

transformations to different parts. The authors experiment with applying augmentation only to the *past* part, and to the *past* and *future* parts separately (so, actually two different separate transformations were applied).

The combination of *pitch+add+reverb* applied only to *past* part (with no transformation of the *future*) showed the best results in the authors' experiments. However, all the experiments with combinations of augmentations were conducted only in this mode (*past* + no augmentation in *future*).

2 Experiments

The experiments in this work can be divided into 2 parts:

- CPC training experiments
- evaluation of the models on downstream tasks

CPC training experiments refer to training different models by modifying their architecture, adding data augmentation, or using different data. However, it is not possible to evaluate and compare CPC models without any additional calculations. CPC's objective is training to predict encoder representations of future timestamps which is not a practical task itself. But we are interested in the audio representations that are learned by the model.

In order to evaluate the feature extractor ability of CPC, its vector representations are used in several downstream tasks which refers to the second part of experiments in this work. Thus, we can compare the CPC models with each other and with some baselines. This would help to evaluate the influence of CPC modifications on its performance as well as to compare them with existing solutions.

The details of all the experiments will be described below. As for software used, all the experiments were conducted using Python 3.10, and its libraries *Pytorch* [17] for writing neural networks, *Pytorch lightning* [18] for training the models and *torchaudio* [19] for all audio transformations.

2.1 CPC training experiments

In the original article about the CPC model [3], the authors do not experiment with the model architecture. However, they conduct ablation studies with varying the number of timestamps in the future to predict ($k = 2, 4, 8, 12, 16$) and different methods of sampling negative examples. They use the accuracy of the downstream phone classification task for evaluating CPC models. They show that predicting a small number of timestamps ($k = 2, 4$) decreases the performance of the model while using $k = 12$

shows the best results (though these results are quite similar to those obtained with $k = 8, 16$). As for negative examples that are necessary for contrastive loss, it was mentioned earlier that they are sampled from the current batch on each step. However, there can be different methods of doing this. The authors compare the following methods:

- constructing the batch from the audios of the same speaker so that all negative examples would be from the speaker of the positive example (*same-speaker*)
- constructing the batch from the audios of different speakers so that negative examples would be sampled from different speakers (*mixed-speaker*)
- negative examples are sampled from the same sequence that a positive example came from (*same-sequence*)

According to their results, it seems that the method of sampling does not influence the results of phoneme classification that much. Accuracy is in a range from 65.6 to 66.5 for all the sampling methods except the method *mixed-speaker* excluding the current sequence from sampling shows the worst result of 57.6.

In this work, there are no experiments similar to described ablation studies. Instead, the best parameters are used.

2.1.1 Training procedure

The basic implementation of the model from the original article by [3] was used.

The AdamW optimizer [20] and a learning rate of 2e-4 are used for training. The learning rate was decreased by a factor of 0.1 if validation loss has not improved for 3 epochs. 2 GPUs were used with a batch of size 8.

In more detail, the training takes place as follows: in one iteration the batch is consisted of 8 audios of size 20480 (shape [8, 1, 20480]). It goes

through the encoder and tensor z of shape [8, 128, 512] is obtained, where 128 is the number of encoded windows of each audio (after downsampling in the encoder), and 512 is the dimensionality of the encoder's hidden state.

Each encoded window of each audio is considered as a positive example z_{pos} (so, there are 8 * 128 of them). Then, negative windows are sampled for each positive one (the number of negative examples is the hyperparameter, here it is 128).

Further, encoder representations of queries z pass through the RNN, and context embeddings c are obtained. They are then used for predicting encodings of the next k timestamps z_{pred} with linear layers.

The optimization problem is reduced to minimizing the cosine of an angle between the vectors for the real encoded windows z_{pos_k} and predicted z_{pred_k}, while maximizing it between negative examples z_{neg_k} and the same predicted windows z_{pred_k}. Loss is calculated for all k and then averaged.

At one iteration, all timestamps of all sequences represent positive examples. For each positive example, 128 negative examples were sampled, similar to [16, 21]. As for the number of timestamps, $k = 12$ is used, similar to [3, 16, 21]. As in the same papers, negative examples are sampled from the sequences of the same speaker as the positive example.

2.1.2 Data

LibriSpeech. The LibriSpeech [22] corpus was used for training CPC models similar to [3]. This is a collection of segmented English audiobooks from the LibriVox project. In the base experiments, *train-clean-100* partition was used for training the models, while *dev-clean* was used as a validation set. *test-clean* subset was used as a dataset for speaker classification in one of the downstream evaluation tasks.

What is more, *train-clean-360* and *train-other-500* subsets were used in the experiments to test whether the increase of the training data can influence the model performance. All train subsets were concatenated into one dataset consisting of 960 hours.

Initially, all audios are of different lengths. During the training of CPC, a random frame of length 20480 is sampled on one epoch for each audio.

Partition	Audios length, hours	# speakers
train-clean-100	100.6	251
train-clean-360	363.6	921
train-other-500	496.7	1166
dev-clean	5.4	40
test-clean	5.4	40

Table 1: Description of the LibriSpeech [22] corpus data partitions.

All the data partitions are described in Table 1 in terms of the number of speakers and total audio length. It is worth mentioning that all these subsets do not overlap by speakers.

Voxceleb2. As it was mentioned earlier, the LibriSpeech dataset consists of very clean audios from audiobooks. However, in most cases, we want to obtain speech representations that can separate the noise from the characteristics of speech and the speaker. In order to check how pre-training on the noisy data can influence the model performance on the downstream tasks, the Voxceleb2 [23] dataset was chosen. This dataset consists of 1.2 million utterances from 6112 speakers. According to the paper, the dataset covers a wide range of speakers of different demographic characteristics, such as ethnicity and age. What is more, the audios are very noisy as they are taken from real interviews of celebrities in very different environments.

The training procedure of model training on this dataset was the same as on LibriSpeech.

2.1.3 Data Augmentation

It was mentioned earlier, [3] shows that data augmentation can improve the performance of the CPC model in terms of phoneme discriminability. This work partly follows the results of this paper. The combination of *pitch+add+reverb* was the best in the authors' experiments. However, the authors focused only on phoneme discriminability. But it seems that pitch modification(shift) could be a too rude transformation if we speak about tasks like speaker identification. So, it was decided to remove *pitch* trans-

formation. Only combination *add+reverb* was used.

In more detail, in experiments with augmenting data for CPC, *add* (adding noise) transformation is just combining the original audio with some randomly selected noise audio from the MUSAN dataset [24] (its noise subset is used here). Noise is added with SNR (signal-to-noise ratio) from 0 to 30 dB. As for *reverb* (room reverb), the effect is added with a randomly selected room scale value from 0 to 100 with other settings set to default. This combination was applied to audio with a probability of 0.7.

As mentioned earlier, 3 data parts can be defined in CPC: *past*, *future-positive*, and *future-negative*. The authors [16] tried only *past* (augmenting only past part) and *past+future* (separate augmentations to past and future) combinations.

In this work, adding/not adding augmentation is added to the ablation studies. Only the *past* part was augmented while the *future* part was left unchanged.

2.1.4 Experiments with architecture

Base architecture. In the original paper [3], CPC model consists of an encoder, an autoregressive recurrent neural network and linear layers for predicting future timestamps. The encoder g_{enc} consists of 5 residual blocks [25] of convolutional layers with kernel sizes [10, 8, 4, 4, 4] and strides [5, 4, 2, 2, 2]. The dimensionality of the output embedding is 512. ReLU activations [26] and Batch Normalization [27] are used between convolutional layers.

After going through the encoder, the signal is sent to the autoregressive part g_{ar} of CPC which is represented by GRU RNN [28] with a 256-dimensional hidden state. The outputs of the GRU are used as the context embeddings c which are then used for predicting K future timestamps for contrastive loss.

This work reuses this architecture in the base experiments. One possible difference is that residual connections are used in the encoder's convolutional layers, though this is not very clear from the paper text whether they use them too. One more detail is that the dimensionality of the hid-

den states is increased by a factor of 2 in each convolutional layer of the encoder.

Further, the models with the base architecture will be called *CPC_noaug* and *CPC_aug* (without and with augmentation correspondingly).

Scaling the number of parameters. There were many attempts to increase the performance of the model on downstream tasks simply by adding extra layers to the ResNet encoder and/or replacing GRU with LSTM. Unfortunately, these experiments did not show any improvement in classification metrics if compared to the results of the model with the base architecture. On the contrary, the quality often decreased for larger networks, even when training on bigger datasets. The possible explanation is that the model overfits the task of predicting the embeddings of the next steps on the training dataset, which entails a loss in quality on downstream tasks. These experiments will be further excluded from the results.

Changing normalization method. Another modification tested was adding audio normalization and changing the normalization method in the encoder. The idea of these changings was taken from the wav2vec2 [1] model which will be described more in the next section. Despite the differences in the training objective and the model architecture, it is pretty similar to the CPC. There are also encoder and context networks and they are trained together with contrastive loss which is almost the same as the InfoNCE loss. The encoder of wav2vec2 is also a convolutional neural network with 7 layers and a hidden size of 512. One important difference is that the audio is normalized before encoding to zero mean and unit variance. What is more, layer normalization is used which was also shown to increase the model performance on the phoneme discrimination tasks in [15].

So, in this part of ablation studies, the waveform is normalized. The encoder consists of 5 convolutional layers with kernel sizes [10, 8, 4, 4, 4] and strides [5, 4, 2, 2, 2]. After each convolution, layer normalization over the channel dimension and GELU [29] activation are performed. The dimensionality of the hidden state of each layer is 512.

The models trained with this encoder will be referred to as *CPC_norm_*.*

Simply increasing the encoder by adding new convolutional layers without downsampling also did not improve the classification metrics, so these results are also excluded.

2.2 CPC evaluation on downstream tasks

To evaluate CPC models and audio representations obtained from them, several downstream tasks were performed. In the original paper [3], the authors evaluate their CPC models on the speaker and phone classification tasks on LibriSpeech *train-clean-100 subset* (the same data that was used for self-supervised training of CPC models). For both tasks, they measure the prediction performance with a simple linear classifier trained on top of the context embeddings c extracted from CPC features. As for the phone prediction task, they use Kaldi [30] for obtaining force-aligned phone sequences and then use 41 phonemes as labels in the phone classification task. In this work, these experiments were also conducted on the same train/val/test splits.

However, evaluating CPC models on the same data it was trained on seems not very correct, and it was done only for comparison of the results with those obtained by [3]. So, several other tasks and datasets were chosen for evaluation. Most of them represent different sequence classification tasks. It was discussed earlier that one can obtain the representation for the whole sequence by pooling (averaging) the features z from the encoder or the context vectors c from RNN. Thus, the pipeline of using the features from CPC in some sequence classification tasks seems quite simple: put the raw waveform of the audio of the arbitrary length to the CPC model, get vectors c or z, average them over the time dimension, send to some classification head that gets one embedding for the whole audio and take its prediction. However, preliminary experiments show that this approach shows poor results. On the other hand, training and inference on separate 1.28s frames works much better. The CPC models are trained on the sampled audios of length 1.28s, which is probably the reason for getting

bad results on the longer sequences. So, the training step of the sequence classification downstream tasks was done on the random frames of length 1.28s sampled from the full audios, and prediction of the class was made on precisely these small sampled frames. While on the validation and testing steps, the audio was cut to the frames of the same length as in the training step, predictions to all the frames of one audio were made and the class label of the full audio was obtained by taking the most frequent prediction.

All the downstream tasks and the datasets used will be discussed below. Table 2 shows the dataset statistics in terms of AdamW optimizer [20] with a learning rate 2e-3 and batch size 64 was used in training all the classifiers. The learning rate was decreased by a factor of 0.1 when validation loss has not decreased for 3 epochs.

dataset	**#classes**	**avg. # ex.**	**std. # ex.**	**min. # ex.**	**max. # ex.**
LBS speaker train	251	91	13	21	134
LBS speaker test	40	20	7	10	36
Voxceleb1	1251	111	78	30	992
Voxceleb1_40_balanced	40	234	24	201	297
SHAL	60	209	0	209	209
SpeechCommands1	30	1703	220	1340	1892
SpeechCommands2	35	2424	789	1256	3250
Samromur Age	2	67197	4945	62252	72142
Samromur Gender	2	40003	10286	29717	50288

Table 2: Description of datasets for downstream tasks in terms of classes distribution. Table shows mean, standard deviation, minimum amd maximum values of the number of examples in each class.

Speaker Identification. The task of speaker identification is to identify the audio speaker among a closed set of alternatives, which can be reduced to a multi-classification problem.

Subsets *train-clean-100* and *test-clean* of LibriSpeech [22] (LBS) were used separately for evaluating how representative the audio features from the CPC model in terms of speaker identity. These two subsets consist of

251 and 41 speakers, respectively.

However, LibriSpeech consists of very clean audios with almost no noise. Therefore, it seems logical to check to what extent models trained on such data can be generalized to noisier data from another domain. As an example of such data, Voxceleb1 [31] (Vox1) corpus was taken. It contains over 100,000 utterances of human speech in English from 1,251 celebrities, extracted from videos uploaded to YouTube. This data is much noisier as it consists of real interviews but not clean audiobooks. What is more, it is very unbalanced. As this dataset turned out to be quite hard, its subsample with only 40 speakers was also added for evaluation (Vox1_40). Speakers are sampled randomly from all the speakers that have from 200 to 300 utterances.

Another question of interest is how much the features from the model can be generalized to speech in another language. It was decided to take the SHALCAS22A [2] (SHAL) dataset for such an experiment. It is the corpus in Chinese that consists of clean human speech recorded in a quiet environment. Clean corpus was chosen deliberately to check the generalization to another language, and not to noisy data. There are audios from 60 speakers in this dataset.

Age and gender identification. Another two downstream tasks are referred to looking at how CPC embeddings can be used for extracting information about the speaker of the audio. Specifically, the CPC features were used to predict the age or gender of the speaker.

As for the gender classification (Gender), Samrómur Icelandic Speech corpus [32] was used. It consists of the recorded human speech from the participants of age 18 to 90.

As for the age classification task (Age), Samrómur Children [33] corpus was used. It is also an Icelandic Speech dataset but from participants of ages 4 to 17. Age was transformed to the binary category by a threshold of 12 years.

[2]SHALCAS22A, a free Chinese Mandarin corpus by Shanghai Acoustics Laboratory, CAS and Wuxi Sandu Intelligent Technology Co., Ltd., 2022; `https://www.openslr.org/138/`

So, both tasks can be represented as a binary classification problem. Both datasets have official splits to train/val/test with no speaker overlap, and exactly these splits were used in the experiments.

Keyword spotting. Another downstream task chosen for evaluation is keyword spotting, which is a word classification problem in this case. This task allows evaluating the performance of CPC features in terms of identifying speech contents, not speaker identity as in the previous tasks.

Datasets Speechcommands1 and Speechcommands2 [34] (Cmd1 and Cmd2) are used for this experiment. They consist of 30 and 35 spoken words, respectively. Each audio represents one word and has a length of 1s. Official train/val/test splits that do not overlap by speakers were used.

Phoneme classification (Phone) As it was mentioned earlier, force-aligned phoneme labels were used for phoneme classification results (the authors [3] made their phoneme labels available, so they were just reused). There are 41 classes in total. In comparison to the previous tasks, this one assumes many labels per sequence. However, the phoneme sequence labels have the same downsampling factor as the base CPC models (one feature vector and one phoneme for each 10 ms), so the algorithm of conducting this experiment is quite simple: take a raw waveform and its phoneme labels, get its CPC features (z or c), and just use corresponding embedding to predict the class of the phoneme (in other words, c_i embedding represents audio's i^{th} phoneme).

2.3 Baselines

To evaluate the performance of the CPC features, several baseline models were used:

MFCC. Probably the most basic and popular baseline in all speech processing tasks is Mel-frequency cepstral coefficients (MFCC). In this work, 256 coefficients, each corresponding to the non-overlapped window of length

160, are used for consistency with the dimensionality and downsampling factor of the method of obtaining CPC features (it is 256 for context embeddings c, the downsampling factor equals 160). Similar to the CPC features, MFCCs are averaged over the time dimension for the sequence classification tasks.

CPC-supervised. Another baseline will be a CPC model that is trained on some downstream tasks in a fully supervised manner (with randomly initialized weights). The base architecture consisting of the ResNet encoder and GRU is used here. Context embeddings were averaged and then used as a speech representation.

Wav2vec2 and HUBERT. Finally, two large transformer models for speech Wav2vec2 [1] and HuBERT [2] are used as baseline feature extractors.

The Wav2Vec2 model has many similarities with the CPC model as it continues its idea. It also consists of the encoder, and context network and is trained with the contrastive objective. However, instead of a simple RNN like GRU in CPC, it uses a multi-layer Transformer [35] as a context network. What is more, the model is trained to distinguish quantized audio representations for masked timestamps among negative examples. This is another important difference of wav2vec2 - it has a quantization module and these quantized (not continuous) vectors are predicted for contrastive loss.

HuBERT has a similar architecture as wav2vec 2.0 but is trained differently. Instead of contrastive learning, HuBERT is trained with a masked language modeling objective similar to BERT [4]. This is possible due to a pre-training step with clusterization of the input to hidden units that are then used as targets in the prediction of randomly masked positions.

In this work, *base* versions of both models are used. Previous research shows that the last hidden state from the context network (Transformer encoder) doesn't need to give the best result as a speech representation in some downstream task if compared to embeddings from other layers [36, 37].

The authors show that some intermediate layers of Transformer or their combinations can give better results for different tasks. The experiments with the best combinations of layers for these models are obviously out of the scope of this work. So, it was decided to use a weighted sum of all the 12 Transformer layers, with training these weights along with training the classification head. This weighted sum is used as sequence embeddings which are also averaged along the time dimension as the CPC vectors and then such representation is used as a feature vector for one 20480-length frame. The dimensionality of the transformer hidden states as well as the averaged embedding is 768.

Wav2vec2 and HuBERT have a different downsampling rate of the input signal in the encoder, and each embedding encodes 25ms of audio. So, evaluation on the phoneme classification task will not be performed for these models, since the labels used encode one phoneme every 10 ms of the signal.

3 Results

3.1 Encoder vs context embeddings

In graph 3, representations obtained from the CPC model with different methods are shown using t-SNE [38] visualization. These representations are shown for audios of 10 speakers from Librispeech *train-clean-100* subset, every color represents one speaker. As seen on the graph 3, features obtained with averaging either z or c are all quite discriminative embeddings for speaker voice characteristics. A similar situation is for *dev-clean* subset.

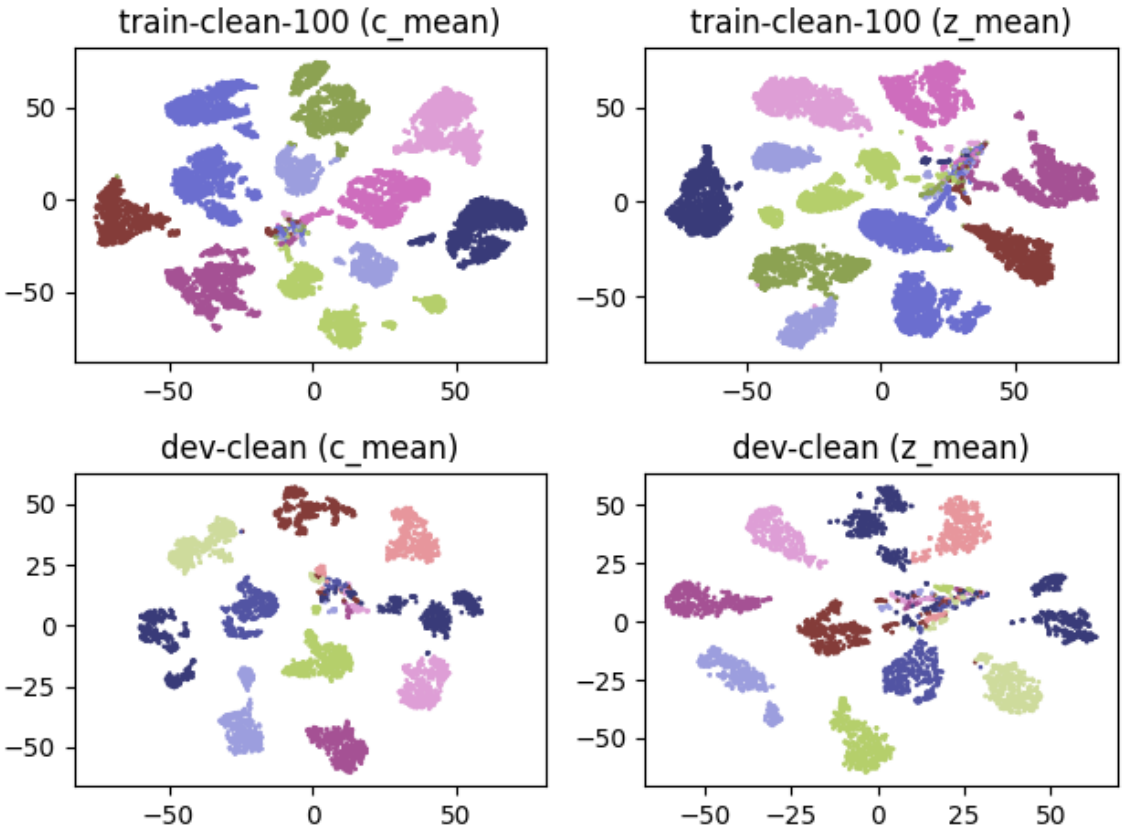

Figure 3: TSNE visualization of *CPC_noaug* features extracted for 10 speakers in train and dev subsets. Color represents the speaker.

For comparison, graph 4 represents a similar picture but for MFCC features. It can be seen that these representations are also discriminative, though a big part of speakers merges into one cluster.

Table 3 shows a comparison of context and encoder representations on downstream tasks using *CPC_noaug* model. Though the authors used only

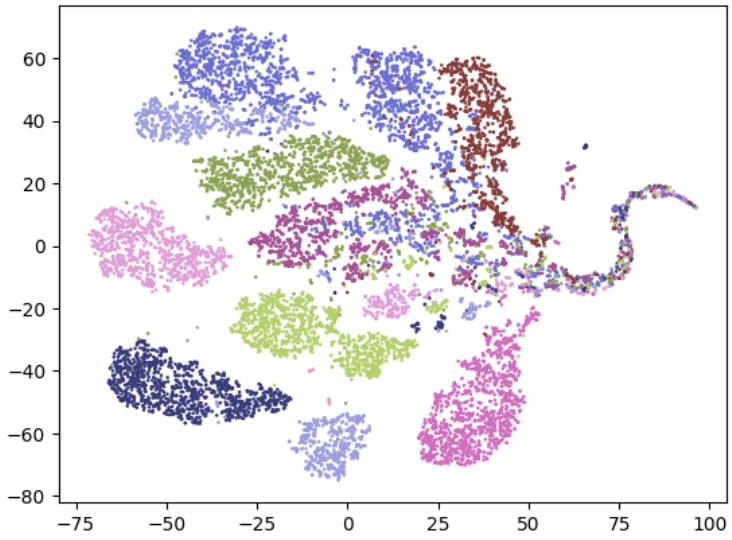

Figure 4: TSNE visualization of MFCC features extracted for 10 speakers in *train-clean-100* subset of LibriSpeech. Color represents the speaker.

context embeddings in the original paper, it can be seen that encoder embeddings z show better results in almost all the classification tasks.

emb	LBS speaker test	LBS speaker train	Phone	Age	Gender	SHAL	Cmd1	Cmd2	Vox1	Vox1_40
c	**1.00**	**1.0**	**0.51**	0.81	**0.95**	0.62	0.57	0.58	0.09	**0.41**
z	0.99	**1.0**	**0.51**	**0.84**	**0.95**	**0.80**	**0.65**	**0.66**	**0.12**	0.40

Table 3: Clasification accuracy on the downstream tasks using encoder and context embeddings from the model *CPC-base-noaug*

3.2 Ablation studies

Table 4 shows the results of ablation studies. Classification results for the speaker identification task on LibriSpeech test-clean and train-clean-100 subsets will be excluded here and further, as all the metrics are in the range of 0.96-1.00 for all the models.

By comparison of *CPC_ aug* model with *CPC_ noaug*, we see that augmentation applied to the data for pretraining the CPC model increases the metrics on downstream tasks by 1 to 9 points depending on the dataset.

model	Phone	Age	Gender	SHAL	Cmd1	Cmd2	Vox1	Vox1_40
Base arhitecture, LibriSpeech 100h								
CPC_noaug	0.51	0.84	0.95	0.80	0.65	0.66	0.12	0.40
+augmentation								
CPC	0.53	0.86	0.96	0.89	0.74	0.73	0.14	0.48
+more training data								
CPC_960h	0.54	0.85	**0.97**	0.86	0.75	0.74	0.14	0.53
CPC_vox2	0.54	0.86	**0.97**	0.86	0.75	0.74	0.16	0.54
Wav2vec2-like encoder								
CPC_norm_100h	0.53	0.84	**0.97**	0.89	0.77	0.75	0.23	0.63
CPC_norm_960h	0.54	**0.87**	**0.97**	0.93	0.78	0.76	0.25	**0.74**
concatting z and c embeddings								
CPC_norm_960h	**0.61**	0.85	**0.97**	**0.95**	**0.86**	**0.84**	**0.32**	0.72
CPC_norm_vox2	0.59	0.84	**0.97**	0.90	0.85	**0.84**	0.30	0.72

Table 4: Ablation studies results. Classification accuracy on downstream tasks. z features from CPC are used.

So, it was decided to use it in all other CPC models.

The more unexpected result is that the increase in the training dataset does not influence the result that much. The models trained on 960 hours of LibriSpeech (*CPC_960h*) and Voxceleb2 (*CPC_vox2*) generally differ by 1-2 points when compared with a similar model trained on only 100 hours of LibriSpeech. The most surprising result is that pretraining on Voxceleb2 does not increase the model performance on the Voxceleb1 speaker classification task. The hypothesis was that pre-training on bigger and noisier data would improve the performance of the model in the task with similar data, however, this hypothesis was not confirmed. These results call into question the possibility of scaling the model with the base architecture in terms of data.

On the other hand, replacing the base ResNet encoder with a wav2vec2-like one increase the metrics in almost all the tasks. This is especially noticeable for the Voxceleb1 speaker classification task, which turned out to be extremely difficult for the CPC models.

The last experiment in this part was to use not only z or c embeddings

but concat them into one embedding of dimensionality 768. This modification increases the metrics on almost all the tasks. This is especially noticeable in the classification of phonemes and spoken commands where accuracy increased by 7-8 points.

3.3 Comparison with Baselines

Table 5 shows the comparison of the best CPC model and baselines.

model	Phone	Age	Gender	SHAL	Cmd1	Cmd2	Vox1	Vox1_40
CPC-supervised	0.80	0.89	0.98	0.88	0.94	0.93	0.29	0.54
MFCC	0.42	**0.85**	0.93	0.48	0.26	0.28	0.12	0.61
HUBERT_base	-	0.84	**0.98**	0.82	**0.97**	**0.96**	**0.67**	**0.93**
Wav2vec2_base	-	**0.85**	**0.98**	0.88	0.96	**0.96**	0.63	0.86
CPC_norm_960h	**0.61**	**0.85**	0.97	**0.95**	0.86	0.84	0.32	0.72

Table 5: Comparison of the CPC model with baselines on downstream tasks. Classification accuracy on downstream tasks. z features from CPC are used.

MFCC turned out to be a strong baseline. Its performance on age and gender classification problems is comparable with the results of the CPC and large models Wav2vec2 and HuBERT. However, the use of pre-trained neural networks becomes more reasonable on other tasks like SHAL and SpeechCommands datasets as well as Voxceleb1 where MFCC gives much worse results.

Performance metrics obtained with CPC features on age and gender classification are almost the same as those obtained with Wav2vec2 and HuBERT. However, as was already mentioned, MFCC copes with the task at the same level which may signal that the use of pre-trained neural network embeddings for the classification tasks on these datasets is redundant.

The CPC model shows worse results on the classification of spoken commands (SpeechCommands1 and SpeechCommands2 datasets) than Wav2vec2 and HuBERT by 10-12 points. A similar situation is with speaker classification on the Voxceleb1 dataset. However, the CPC outperforms the large

models on the classification of Chinese speakers (SHAL) by 7-13 points. This is a rather interesting result, showing that the CPC features can be applied even under conditions of a domain shift (different language). Nevertheless, it is worth mentioning that the SHAL dataset is a corpus of clean recorded speech almost without any noise (so, it is similar to LibriSpeech in this sense). If compared to Wav2vec2 and HuBERT, the CPC model performance in the classification of noisy data (Voxceleb1) is much worse, even despite the augmentation in the pre-training.

model	**Total**	**Encoder**	**Context**	**Prediction**
wav2vec2|HUBERT (base)	94.4			
CPC	4.6	2.5	0.6	1.7
CPC_norm	7.4	5.3	0.6	1.7

Table 6: The number of model parameters.

Table 6 demonstrates a comparison of the CPC models with the large models Wav2vec2 and HuBERT in terms of the number of parameters. The number of parameters in CPC models is an order of magnitude lower than in the large models.

3.4 Interactive Speaker Identification

Another additional task in which CPC representations have been used is the speaker identification task with the Reinforcement Learning (RL) method of interactive speaker recognition proposed by [39]. The model is trained to identify a speaker from a known set by several words which it learns to request in a way that will maximize speaker identification accuracy. The baseline features here are x-vectors [40] which are embeddings from a pre-trained convolutional neural network.

It can be seen from Table 7 that the use of CPC features significantly improves identification accuracy by almost 20 points.

model	Random	RL agent
x-vectors	0.75	0.91
CPC_norm_960h	**0.945**	**0.99**

Table 7: Accuracy of speaker identification with requesting 2 words chosen randomly or with the use of trained RL agent. The total number of speakers is 20.

3.5 Discussion

As it was seen from ablation studies, with the help of small changes in the architecture, augmentation, and an increase in the dataset, it was possible to increase the performance of CPC on all downstream tasks. However, even though the classification accuracy has improved, Voxceleb1 turned out to be the most problematic dataset for downstream tasks as it is very noisy and unbalanced data. However, even for 40 speakers with approximately the same number of utterances, the model performs worse than for the dataset SHAL with a larger number of speakers and a different language. This tells us that most likely the model still has problems separating noise from speaker and speech features.

Large models cope with classification on the Voxceleb1 dataset much better. However, as has been shown, their size exceeds the CPC size by an order of magnitude. The main part of the parameters of these models is in the context network, which is represented by Transformer, while their encoder and CPC encoder are quite similar in both size and architecture. Moreover, representations from large models were taken from all layers of the transformer, and not from the CNN model. Thus, it is logical to assume that replacing a simple GRU RNN network with a Transformer could significantly improve the results. However, then the model would lose its main advantage - small size. In the existing literature, it was not possible to find examples of using the Transformer architecture in models with a training objective of CPC - in fact, the following self-supervised models for Speech Processing moved towards vector quantization and prediction of masked positions in the sequence rather than predicting vectors of future steps (like VQ-wav2vec, Wav2vec2, HuBERT). Therefore, it is not obvious

how much such a replacement would give an increase in results. In addition, it would be interesting to experiment with types of augmentation of pretraining data and changing and increasing the architecture not just by adding layers without downsampling.

Despite the previous considerations, it is still worth noting that on other tasks related to the classification of speakers or their features, the CPC models show comparative or even better results than Wav2vec2 and HuBERT. Thus, we can conclude that the CPC model can compete with large models, although not on all data and tasks. Moreover, model pretraining is practically useful, since the use of CPC features on new data in downstream tasks is not very different from fully training the model with the same architecture entirely on the new dataset in most cases.

Conclusion

In this work, the method of contrastive predictive coding was described in detail. The model from the original paper was reproduced and experiments with CPC features on the base downstream tasks were conducted.

Then, additional downstream tasks were described. They evaluate representations extracted from the CPC from different angles: information about the speaker and speaker characteristics, the content of the speech; the possibility of domain shift in terms of language and noise.

Further, there were ablation studies on adding augmentation, data normalization, changing the architecture of the model, and increasing the dataset for pretraining. During these experiments, the model performance was improved on all downstream tasks.

Finally, the results of the best CPC model were compared with baselines including large pretrained models. It has been shown that the CPC model can compete with large models on some tasks despite its small size. Thus, it was concluded that in conditions of limited resources, the use of the CPC can be justified. However, the CPC model works worse in some cases. On noisier datasets, the CPC model features are worse than the features of the large models, which is most likely due to the big difference in the size of the models.

References

[1] A. Baevski, H. Zhou, A. Mohamed, and M. Auli, "wav2vec 2.0: A framework for self-supervised learning of speech representations," 2020.

[2] W.-N. Hsu, B. Bolte, Y.-H. H. Tsai, K. Lakhotia, R. Salakhutdinov, and A. Mohamed, "Hubert: Self-supervised speech representation learning by masked prediction of hidden units," 2021.

[3] A. v. d. Oord, Y. Li, and O. Vinyals, "Representation learning with contrastive predictive coding," 2018.

[4] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," 2019.

[5] A. Radford and K. Narasimhan, "Improving language understanding by generative pre-training," 2018.

[6] P. H. Le-Khac, G. Healy, and A. F. Smeaton, "Contrastive representation learning: A framework and review," *IEEE Access*, vol. 8, pp. 193907–193934, 2020.

[7] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," 2013.

[8] T. Chen, S. Kornblith, M. Norouzi, and G. Hinton, "A simple framework for contrastive learning of visual representations," 2020.

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

[10] A. Baevski, S. Schneider, and M. Auli, "vq-wav2vec: Self-supervised learning of discrete speech representations," 2020.

[11] S. Chopra, R. Hadsell, and Y. LeCun, "Learning a similarity metric discriminatively, with application to face verification," in *2005 IEEE*

Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05), vol. 1, pp. 539–546 vol. 1, 2005.

[12] R. Hadsell, S. Chopra, and Y. LeCun, "Dimensionality reduction by learning an invariant mapping," in *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*, vol. 2, pp. 1735–1742, 2006.

[13] M. Gutmann and A. Hyvärinen, "Noise-contrastive estimation: A new estimation principle for unnormalized statistical models," in *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics* (Y. W. Teh and M. Titterington, eds.), vol. 9 of *Proceedings of Machine Learning Research*, (Chia Laguna Resort, Sardinia, Italy), pp. 297–304, PMLR, 13–15 May 2010.

[14] R. Jozefowicz, O. Vinyals, M. Schuster, N. Shazeer, and Y. Wu, "Exploring the limits of language modeling," 2016.

[15] J. Kahn, M. Riviere, W. Zheng, E. Kharitonov, Q. Xu, P. Mazare, J. Karadayi, V. Liptchinsky, R. Collobert, C. Fuegen, T. Likhomanenko, G. Synnaeve, A. Joulin, A. Mohamed, and E. Dupoux, "Libri-light: A benchmark for ASR with limited or no supervision," in *ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, IEEE, may 2020.

[16] E. Kharitonov, M. Rivière, G. Synnaeve, L. Wolf, P.-E. Mazaré, M. Douze, and E. Dupoux, "Data augmenting contrastive learning of speech representations in the time domain," 2020.

[17] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala, "Pytorch: An imperative style, high-performance deep learning library," in *Advances in Neural Information Processing Systems* (H. Wallach, H. Larochelle, A. Beygelzimer,

F. d'Alché-Buc, E. Fox, and R. Garnett, eds.), vol. 32, Curran Associates, Inc., 2019.

[18] W. Falcon and The PyTorch Lightning team, "PyTorch Lightning," Mar. 2019.

[19] Y.-Y. Yang, M. Hira, Z. Ni, A. Chourdia, A. Astafurov, C. Chen, C.-F. Yeh, C. Puhrsch, D. Pollack, D. Genzel, D. Greenberg, E. Z. Yang, J. Lian, J. Mahadeokar, J. Hwang, J. Chen, P. Goldsborough, P. Roy, S. Narenthiran, S. Watanabe, S. Chintala, V. Quenneville-Bélair, and Y. Shi, "Torchaudio: Building blocks for audio and speech processing," *arXiv preprint arXiv:2110.15018*, 2021.

[20] I. Loshchilov and F. Hutter, "Decoupled weight decay regularization," 2019.

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

[22] V. Panayotov, G. Chen, D. Povey, and S. Khudanpur, "Librispeech: An asr corpus based on public domain audio books," in *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 5206–5210, 2015.

[23] J. S. Chung, A. Nagrani, and A. Zisserman, "Voxceleb2: Deep speaker recognition," in *INTERSPEECH*, 2018.

[24] D. Snyder, G. Chen, and D. Povey, "Musan: A music, speech, and noise corpus," 2015.

[25] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," 2015.

[26] A. F. Agarap, "Deep learning using rectified linear units (relu)," *arXiv preprint arXiv:1803.08375*, 2018.

[27] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," 2015.

[28] K. Cho, B. van Merrienboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, "Learning phrase representations using rnn encoder-decoder for statistical machine translation," 2014.

[29] D. Hendrycks and K. Gimpel, "Gaussian error linear units (gelus)," 2020.

[30] 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 *IEEE 2011 Workshop on Automatic Speech Recognition and Understanding*, IEEE Signal Processing Society, Dec. 2011. IEEE Catalog No.: CFP11SRW-USB.

[31] A. Nagrani, J. S. Chung, and A. Zisserman, "Voxceleb: a large-scale speaker identification dataset," in *INTERSPEECH*, 2017.

[32] D. E. Mollberg, Ó. H. Jónsson, S. Þorsteinsdóttir, S. Steingrímsson, E. H. Magnúsdóttir, and J. Gudnason, "Samrómur: Crowd-sourcing data collection for Icelandic speech recognition," in *Proceedings of the 12th Language Resources and Evaluation Conference*, (Marseille, France), pp. 3463–3467, European Language Resources Association, May 2020.

[33] C. Mena, M. Borsky, D. E. Mollberg, S. F. Guðmundsson, S. Hedström, R. Pálsson, Ó. Helgi, Jónsson, S. Þorsteinsdóttir, J. V. Guðmundsdóttir, E. H. Magnúsdóttir, R. Þórhallsdóttir, and J. Gudnason, "Samrómur Children Icelandic Speech 21.09," Reykjavik University: Language and Voice Lab, 2021.

[34] P. Warden, "Speech commands: A dataset for limited-vocabulary speech recognition," 2018.

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

[36] A. Pasad, J.-C. Chou, and K. Livescu, "Layer-wise analysis of a self-supervised speech representation model," 2022.

[37] L. Pepino, P. Riera, and L. Ferrer, "Emotion recognition from speech using wav2vec 2.0 embeddings," 2021.

[38] L. van der Maaten and G. Hinton, "Visualizing data using t-SNE," *Journal of Machine Learning Research*, vol. 9, pp. 2579–2605, 2008.

[39] M. Seurin, F. Strub, P. Preux, and O. Pietquin, "A machine of few words – interactive speaker recognition with reinforcement learning," 2020.

[40] D. Snyder, D. Garcia-Romero, G. Sell, D. Povey, and S. Khudanpur, "X-vectors: Robust dnn embeddings for speaker recognition," in *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 5329–5333, 2018.