Self-supervised learning applied to speaker and language recognition

Theo Lepage
(supervisor: Reda Dehak)

Technical Report n^ooutput, June 2021
revision

Deep learning is considered as a major breakthrough in the field of machine learning. However, training large models in a supervised way requires a massive amount of labeled samples. This represents a bottleneck to the development of such methods especially for speech related tasks. Self-supervised learning recently gained in popularity as it allows pre-training models without requiring manually annotated labels. This report aims at studying how it could be applied to speaker and language recognition through a data-efficient evaluation.

L'apprentissage de réseaux de neurones profonds est considéré comme une avancée majeure dans le domaine de l'apprentissage automatique. Cependant, entraîner ces modèles de manière supervisée nécessite une quantité massive d'échantillons étiquetés. Cela représente un frein au développement de ces méthodes, en particulier en ce qui concerne les tâches liées à la reconnaissance de la parole. L'apprentissage auto-supervisé a récemment gagné en popularité car il permet de pré-entraîner des modèles sans nécessiter d'étiquettes annotées manuellement. Ce rapport vise à étudier comment cela pourrait être appliqué à la reconnaissance du locuteur et du langage à travers une évaluation basée sur leur consommation de données.

Keywords
Speaker recognition, Language recognition, Self-supervised learning, Data-efficient deep learning, Mutual information, Representation learning

Laboratoire de Recherche et Développement de l'EPITA
14-16, rue Voltaire – FR-94276 Le Kremlin-Bicêtre CEDEX – France
Tél. +33 1 53 14 59 22 – Fax. +33 1 53 14 59 13
theo.lepage@epita.fr – http://www.epita.fr/

Copying this document

Contents

Acknowledgments

I would like to thank my supervisor Reda Dehak for the help he provided me during the semester and his benevolence. Furthermore, I thank all the teachers, researchers and students I met throughout my studies and especially since my enrollment to RDI, EPITA's research class.

This semester has been an enriching experience and an incredible opportunity to learn, experiment and discover the world of scientific research.

Chapter 1

Introduction

Recent breakthroughs in the field of machine learning are due to the rise of deep learning through neural networks. The vast majority of these models are considered discriminative as they are trained to match a set of input values to their corresponding label. This behavior make them inherently supervised which is the reason why they rely on a supervised training phase beforehand. Furthermore, as objective tasks become harder over the years, recent models tend to increase their capacity by increasing their depth and thus their number of trainable parameters.

As a consequence, larger models require a greater amount of labeled samples during the training to keep generalizing beyond their training set and avoid what is commonly referred to as overfitting. The problem is that labeling data is expensive, tedious and slow. Furthermore, manual labeling is not scalable to the amount of data available today. Moreover, the risk is to create a biased model by training on data not representative of the real life application. Therefore, supervised learning is now considered as a bottleneck to build more intelligent systems. For instance, a language recognition system would theoretically need labeled samples for each of the 7,000 languages spoken in the world, which is not feasible.

Over the last few years, several techniques referred to as self-supervised learning have been proposed to solve this very specific but crucial issue. These methods aims at training neural networks without manually annotated labels. They rely on the input signal itself and sampling strategies to identify patterns in training data without the need of a supervisory signal. Afterward, the representations learned can be used to solve the original task with less labeled samples.

Our objective is to apply these techniques to speaker and language recognition which are also concerned by the lack of large labeled datasets. The first chapter 2 is dedicated to a review of existing self-supervised learning techniques that guided our work. Mutual information, which is a key concept at the base of many self-supervised methods, will be presented in details in Chapter 3. Then, in Chapter 4, we are going to approach the different implementation details. Finally, the results of our experiments will be reported in Chapter 5 from a data-efficient point of view.

Chapter 2

Self-supervised learning

2.1 Motivation

Supervised learning is a bottleneck when it comes to building large models consuming a lot of labeled data. On the contrary, self-supervised learning relies on supervisory signals generated from the data itself. The key principle is to train a model on a pretext task, that does not require any kind of human supervision, and use the representations learned for a different downstream task through transfer learning for instance. It is noteworthy that the difference between unsupervised and self-supervised learning is subtle as the latter is essentially neural networks trained in a supervised way but with labels determined in an unsupervised fashion.

Many researchers believe that self-supervised learning is the next step towards building more intelligent models. It is also considered as a solution to the inability of artificial intelligence to generalize with only a few examples which remains an unsolved problem. As opposed to humans, who are able to drive a car after 20 hours of driving lessons in France, the best self-driving cars still require thousands of hours of training data. The intuition is that self-supervised learning is the key to build a kind of background knowledge that current systems usually lack. This is motivated by two assets of self-supervised learning, the ability to train on large training sets and the use of generalist training objectives that can later be useful for a variety of downstream tasks.

In the following subsection we are going to see how self-supervised learning has been applied to audio signals for speech, speaker and language recognition.

2.2 Related work

2.2.1 Contrastive Predictive Coding (CPC)

Contrastive Predictive Coding (CPC) [van den Oord et al. (2019)] aims to extract useful representations from high-dimensional data in a unsupervised way. The model predicts the next input chunk of a sequence using the encoded representation of the input and the current context. CPC is able to learn interesting and high-level information for various downstream tasks such as speech recognition, computer vision, text and reinforcement learning. It relies on the following concepts:

- **Contrastive**: the model is trained to discern between positive and negative samples, in this case with a probabilistic contrastive loss.
- **Predictive**: the model has to predict future patterns given the current context, which is a common strategy for unsupervised learning.
- **Coding**: predictions are performed in a latent space by encoding input data first as opposed to predicting high-dimensional data directly.

The objective of the model is to encode the information shared between different parts of the input signal. Indeed, low-level and local information such as noise must be discarded in the feature vectors. Furthermore, when predicting further in the future, the amount of shared information is decreasing which enables capturing *slow-features* that span many time steps (e.g. story line in books or objects in images).

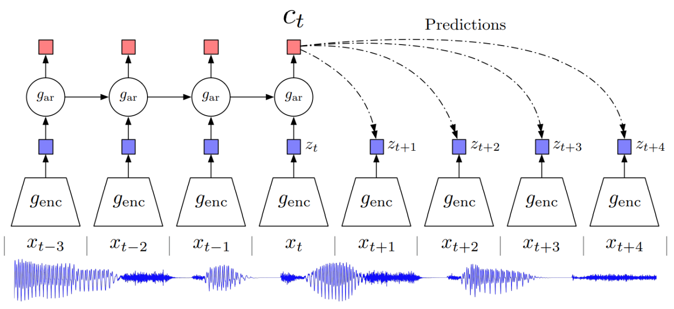


Figure 2.1 – CPC model architecture for audio signals.

As shown on Figure 2.1, input chunks x_t are first fed to an encoder and the generated features vectors z_t are summarized by an autoregressive model to produce the context c_t. For each time step t, the similarity between the prediction and the actual vector is denoted $f_k\left(x_{t+k}, c_t\right)$ and expressed by Eq. 2.1. During training, we consider a set $X = \{x_1, \ldots x_N\}$ of N random samples containing one positive sample and $N-1$ negative samples. The encoder and the autoregressive model are trained jointly to optimize a loss based on Noise-Contrastive Estimation [Gutmann and arinen (2010)] and defined by Eq. 2.2.

$$f_k\left(x_{t+k}, c_t\right) = \exp\left(z_{t+k}^T W_k c_t\right) \tag{2.1}$$

$$\mathcal{L}_{\text{NCE}} = -\underset{X}{\mathbb{E}}\left[\log \frac{f_k\left(x_{t+k}, c_t\right)}{\sum_{x_j \in X} f_k\left(x_j, c_t\right)}\right] \tag{2.2}$$

Optimizing this loss will result in $f_k\left(x_{t+k}, c_t\right)$ estimating the density ratio $\frac{p(x_{t+k}|c_t)}{p(x_{t+k})}$ and will maximize a lower bound on mutual information between x_{t+k} and c_t.

This method was successfully applied to speaker recognition by training a linear classifier on top of its representations. For a subset of 251 speakers from LibriSpeech [Vassil Panayotov and Khudanpur (2015)], CPC achieves the accuracy of 97.4% which is really close to the fully supervised networks reaching 98.5%. Finally, a t-SNE visualization showed that the embeddings are really efficient to discriminate speaker voice characteristics.

2.2.2 Local and Global Info Max (LIM / GIM)

Local Info Max (LIM) [Ravanelli and Bengio (2019a)] is a technique that aims to maximize mutual information between representations of speech sampled from the same sentence. The idea is to learn an embedding that contains the speaker identity. It relies on a local sampling strategy and on an encoder-discriminator architecture similarly to Generative Adversial Networks (GANs) as shown on Figure 2.2. **Global Info Max (GIM)** is equivalent but averages encoded frames within a long chunk to work on larger context windows. The idea is to learn complementary higher-level information to those obtained with LIM.

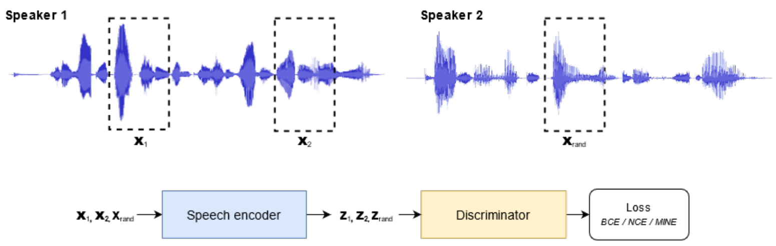


Figure 2.2 – The LIM architecture.

The encoder, based on SincNet [Ravanelli and Bengio (2019b)], converts raw speech waveform to feature vectors and the discriminator is fed by two representations from the same sentence (the anchor z_1 and the positive sample z_2 from the joint distribution) and by a representation from a different sentence (the negative sample z_{rnd} from the product of marginal distributions). They are both jointly trained to maximize the divergence between the two distributions as the aim is to maximize the mutual information between two speech representations sampled from the same sentence (z_1 and z_2). Furthermore, differently to GANs, they are not adversarial and must cooperate (*max-max* game instead of a *min-max* game).

Mutual information should capture the speaker identity as it is the constant factor characterizing chunks of speech of the same sentence as opposed to phonemes. The assumption is that each sentence contains a single speaker and that two random sentences likely belong to different speakers. Thus, to perform the speaker identification task, a simple multilayer perceptron (MLP) is used on top of the encoder to output a set of posterior probabilities over the targeted speakers.

2.2.3 wav2vec, vq-wav2vec and wav2vec 2.0

wave2vec 2.0 [Baevski et al. (2020b)] is a framework for self-supervised learning of representations from raw audio data. The previsous versions, **wav2vec** [Schneider et al. (2019)] and **vq-wav2vec** [Baevski et al. (2020a)], are very similar to CPC (2.2.1) but the latter introduced the use of a quantization module to produce targets for the contrastive task and Transformers [Vaswani et al. (2017)] to generate contextualized representations. The difference is that **wav2vec 2.0** builds context representations over continuous speech representations.

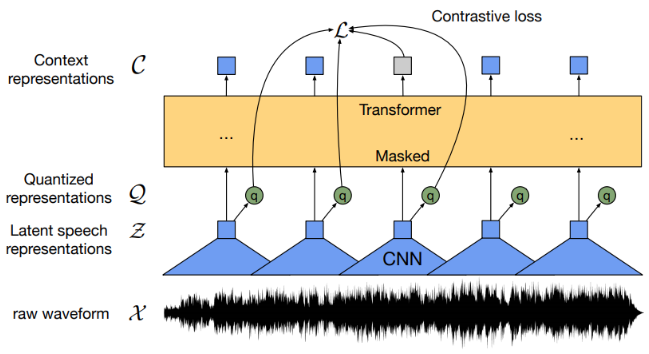


Figure 2.3 – wave2vec 2.0 model architecture.

As shown on Figure 2.3, a multi-layer convolutional neural network is used to produce features, in a latent space, from input raw audio. These speech representations are then randomly masked and then fed to a Transformer to generate contextualized representations. During self-supervised learning the output of the feature encoder is discretized with a quantization module relying on a Gumbel softmax. These quantized representations are used as targets in the contrastive task where the model has to distinguish the expected target from distractors.

The quantization module consists in choosing among $V = 320$ entries of size $d = 120$ from $G = 2$ codebooks (dictionnaries) and concatenating the resulting vectors. Using a straight-through estimator [Bengio et al. (2013)], the Gumbel softmax [Jang et al. (2017)] enables choosing discrete entries in a fully differentiable way as operations such as argmax or argmin are not differentiable. The probability of choosing the v-th entry of codebook g is defined as:

$$p_{g,v} = \frac{\exp\left(l_{g,v} + n_v\right)/\tau}{\sum_{k=1}^{V} \exp\left(l_{g,k} + n_k\right)/\tau}, \tag{2.3}$$

where $\mathbf{l}$ is a mapping of the feature encoder output, $n = -\log(-\log(u))$ and u are uniform samples from $\mathcal{U}(0,1)$ and τ is the temperature.

Regarding objective functions, $\mathcal{L}_m$ requires the model to identify the true quantized speech representation for a time step within a set of $K = 100$ distractors. Note that $\operatorname{sim}(\mathbf{a},\mathbf{b})$ is the cosine similarity and that κ is an hyperparameter.

$$\mathcal{L}_m = -\log \frac{\exp\left(\operatorname{sim}\left(\mathbf{c}_t, \mathbf{q}_t\right)/\kappa\right)}{\sum_{\tilde{\mathbf{q}}\sim\mathbf{Q}_t} \exp\left(\operatorname{sim}\left(\mathbf{c}_t, \tilde{\mathbf{q}}\right)/\kappa\right)} \tag{2.4}$$

The model was originally applied to speech recognition and outperforms the previous state of the art while using 100 times less labeled data. However it is theoretically possible to use a similar architecture for speaker and language recognition by changing the way negatives are sampled (across different utterances and not within the same utterance).

2.2.4 PASE and PASE+

PASE (Problem-Agnostic Speech Encoder) [Pascual et al. (2019)] relies on an encoder jointly trained by multiple workers solving different self-supervised tasks. These workers, cooperating to discover meaningful speech representations, make the proposed approach problem-agnostic for speech related tasks. The approach requires consensus across tasks which implies learning general and transferable features.

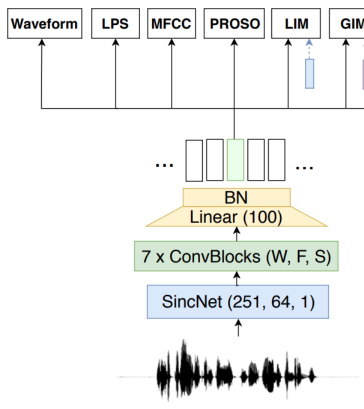


Figure 2.4 – PASE model architecture.

As shown by Figure 2.4, the feature vector generated by the encoder is fed to each of the self-supervised modules which can be grouped into two types: regressors and discriminators. All of them contribute to add prior knowledge into the encoder. During training, regressors such as **Waveform**, **Log Power Spectrum (LPS)**, **Mel-Frequency Cepstral Coefficients (MFCC)** and **Prosody** aim to minimize the mean squared error (MSE) between network predictions and features extracted from input audio signal with Python libraries for audio analysis such as librosa. Discriminators such as **Local Info Max (LIM)** (2.2.2), **Global Info Max (GIM)** (2.2.2) and **Sequence Predictive Coding (SPC)** (2.2.1) learn higher level of abstraction by maximizing mutual information between representations from the same context.

The authors have shown that each worker contribute to the final accuracy even if some are more application-dependent, for instance **LIM** and **GIM** are particularly helpful for speaker identification. Training a multilayer perceptron (MLP) classifier by using PASE as a feature extractor, achieves a 97.3% classification accuracy on VCTK dataset. By training PASE jointly with the classifier the accuracy reaches 99.3% when MFCC and FBANK features achieves 96.9% and 98.4% respectively. Furthermore, PASE even outperforms other systems in difficult acoustic conditions.

PASE+ [Ravanelli et al. (2020)] was later introduced and works better in noisy and reverberant environments. The main differences with the previous version are the use of a speech distortion module to add various random distortions to the input signal ; a new encoder architecture and several new regressors working on longer context. Compared to the previous version, these changes led to substantial relative improvement of 9.5% in clean scenario and of 17.7% in noisy conditions.

2.3 Overview of different learning strategies

All these methods share a common objective of producing high level representations of sound signals in a self-supervised fashion and can be grouped into the following two categories:

- models acting as regular auto-encoders and relying on existing audio processing techniques to produce targets as performed by regressors modules of PASE (2.2.4) ;
- models relying on sampling strategies to solve a contrastive task in order to maximize mutual information between similar audio representations which is the strategy of all other methods (2.2.1, 2.2.2, 2.2.3) and the discriminators modules of PASE (2.2.4).

In the next section we are going to focus on contrastive tasks by understanding what is the role of mutual information and how the optimization of specific objective functions results in the ability to learn meaningful representations.

Chapter 3

The role of mutual information

3.1 Definition

Mutual information is used to estimate the statistical dependence between two random variables by capturing complex non-linear relationships, as opposed to metrics such as correlation. For discrete distributions mutual information is defined by Eq 3.1.

$$MI\left(z_1, z_2\right) = \sum_{z_1} \sum_{z_2} p\left(z_1, z_2\right) \log\left(\frac{p\left(z_1, z_2\right)}{p\left(z_1\right) p\left(z_2\right)}\right) = D_{KL}\left(p\left(z_1, z_2\right) \| p\left(z_1\right) p\left(z_2\right)\right) \tag{3.1}$$

D_{KL} is the *Kullback-Leibler* divergence, often abbreviated *KL*, between the joint distribution $p(z_1, z_2)$ and the product of marginals $p(z_1)p(z_2)$. MI is minimized when the two variables are independent, in which case $MI = 0$, and is maximized when they contain the same information.

3.2 Objective functions maximizing mutual information

Mutual information is a promising tool for learning meaningful representations in an unsupervised way. However, computing directly the mutual information between two random variables is hard in high dimensional spaces. Different objective functions exist to estimate this metric through neural networks. The following formulas assume that z_1 and z_2 are sampled from the same context, which z_{rand} does not belong to, and that g is a neural network.

- **Binary cross-entropy (BCE)**: this metric estimates the *Jensen-Shannon* divergence rather than the *Kullback–Leibler (KL)* divergence. Therefore, this loss does not optimize the exact definition of the mutual information based on the *KL* divergence. However, it is bounded making it easier to optimize.

$$L(z_1, z_2, z_{rand}, \Theta) = \mathbb{E}_{X_p}\left[\log\left(g\left(z_1, z_2\right)\right)\right] + \mathbb{E}_{X_n}\left[\log\left(1 - g\left(z_1, z_{rand}\right)\right)\right] \tag{3.2}$$

- **Mutual Information Neural Estimation (MINE)** [Belghazi et al. (2018)]: this metric computes mutual information by relying on a lower bound expressed as the *Donsker-Varadhan* representation of the *KL* divergence.

$$L(z_1, z_2, z_{rand}, \Theta) = \mathbb{E}_{X_p}\left[g\left(z_1, z_2\right)\right] - \log\left(\mathbb{E}_{X_n}\left[e^{g(z_1, z_{rand})}\right]\right) \tag{3.3}$$

- **Noise Contrastive Estimation (NCE)** [Gutmann and arinen (2010)]: this metric has been proven to maximize a lower bound of mutual information in van den Oord et al. (2019).

$$L(z_1, z_2, z_{rand}, \Theta) = \mathbb{E}_x\left[g\left(z_1, z_2\right) - \log\left(e^{g(z_1, z_2)} + \sum_{x_n} e^{g(z_1, z_{rand})}\right)\right] \tag{3.4}$$

All of these objective functions were compared in LIM original publication [Ravanelli and Bengio (2019a)] and the best performance was achieved with the standard binary cross-entropy loss. Moreover, they were all outperforming the triplet loss which implies that mutual information is more meaningful in this case than a simple euclidean or cosine distance.

We conducted several experiments to reproduce these results and assess the ability of the different objective functions to converge. We used the same framework detailed in section 5 and reported the evolution of the loss on Figure 3.1. In most cases the model struggle to converge especially for LIM tasks which are harder to solve than for CPC. Indeed, CPC has more compute capability and has to pick the correct sample in the mini-batch instead of outputting a scalar value according to only two samples.

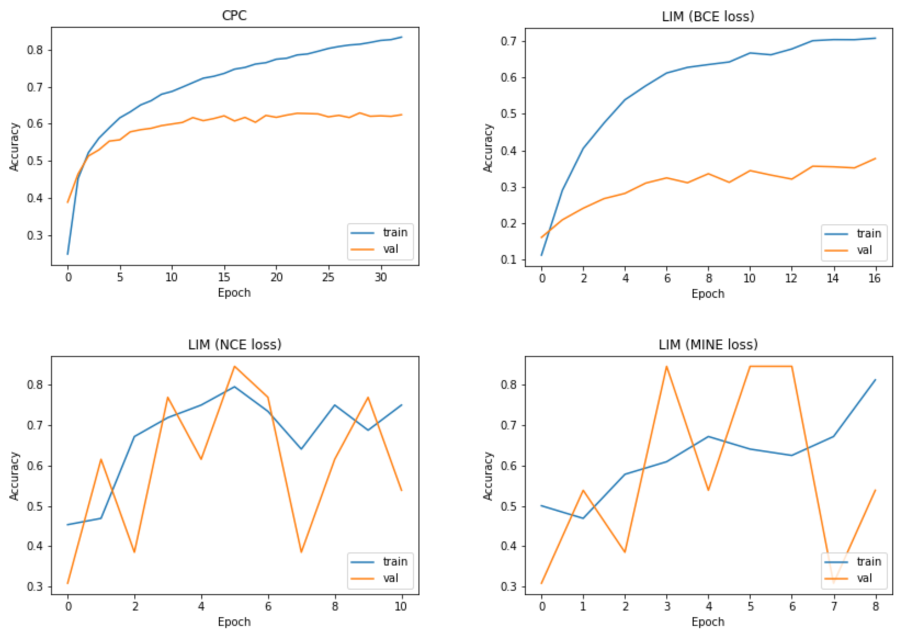


Figure 3.1 – Evolution of different loss functions aiming at maximizing mutual information.

3.3 Noise-Contrastive Estimation loss

In this subsection, we are going to show why minimizing the NCE loss results in the mutual information between x_{t+k} and c_t being maximized.

$$\mathcal{L}_{\text{NCE}} = -\mathop{\mathbb{E}}_{X}\left[\log \frac{f_k\left(x_{t+k}, c_t\right)}{\sum_{x_j \in X} f_k\left(x_j, c_t\right)}\right] \tag{3.5}$$

According to van den Oord et al. (2019) the optimal solution for $f_k\left(x_{t+k}, c_t\right)$ is $\frac{p(x_{t+k}|c_t)}{p(x_{t+k})}$.

$$\mathcal{L}_{\text{NCE}}^{\text{opt}} = -\mathop{\mathbb{E}}_{X}\log\left[\frac{\frac{p(x_{t+k}|c_t)}{p(x_{t+k})}}{\frac{p(x_{t+k}|c_t)}{p(x_{t+k})} + \sum_{x_j \in X_{\text{neg}}} \frac{p(x_j|c_t)}{p(x_j)}}\right] \tag{3.6}$$

$$= \mathop{\mathbb{E}}_{X}\log\left[1 + \frac{p\left(x_{t+k}\right)}{p\left(x_{t+k} \mid c_t\right)} \sum_{x_j \in X_{\text{neg}}} \frac{p\left(x_j \mid c_t\right)}{p\left(x_j\right)}\right] \tag{3.7}$$

$$\approx \mathop{\mathbb{E}}_{X}\log\left[1 + \frac{p\left(x_{t+k}\right)}{p\left(x_{t+k} \mid c_t\right)}(N-1) \mathop{\mathbb{E}}_{X_{\text{neg}}} \frac{p\left(x_j \mid c_t\right)}{p\left(x_j\right)}\right] \tag{3.8}$$

We can compute the term $\mathop{\mathbb{E}}_{X_{\text{neg}}} \frac{p(x_j|c_t)}{p(x_j)}$ as $\frac{p(x_j|c_t)}{p(x_j)}$ is a ratio of two continuous probability density functions.

$$\mathop{\mathbb{E}}_{X_{\text{neg}}} \frac{p\left(x_j \mid c_t\right)}{p\left(x_j\right)} = \int_{X_{\text{neg}}} \frac{p(x_j \mid c_t)}{p(x_j)} p(x_j) dx_j = \frac{1}{p(c_t)} \int_{X_{\text{neg}}} p(x_j, c_t) dx_j = \frac{p(c_t)}{p(c_t)} = 1 \tag{3.9}$$

Furthermore, we know that $\frac{p(x_{t+k})}{p(x_{t+k}|c_t)} \leq 1$ because $p\left(x_{t+k}\right) \leq p\left(x_{t+k} \mid c_t\right)$.

$$\mathcal{L}_{\text{NCE}}^{\text{opt}} = \mathop{\mathbb{E}}_{X}\log\left[1 + \frac{p\left(x_{t+k}\right)}{p\left(x_{t+k} \mid c_t\right)}(N-1)\right] \geq \mathop{\mathbb{E}}_{X}\log\left[\frac{p\left(x_{t+k}\right)}{p\left(x_{t+k} \mid c_t\right)} N\right] \tag{3.10}$$

$$\mathop{\mathbb{E}}_{X}\log\left[\frac{p\left(x_{t+k}\right)}{p\left(x_{t+k} \mid c_t\right)} N\right] = -\mathop{\mathbb{E}}_{X}\log\left[\frac{p\left(x_{t+k} \mid c_t\right)}{p\left(x_{t+k}\right)}\right] + \mathop{\mathbb{E}}_{X}\log\left[N\right] \tag{3.11}$$

$$= -MI(x_{t+k}, c_t) + log(N) \tag{3.12}$$

Therefore we have $MI\left(x_{t+k}, c_t\right) \geq \log(N) - \mathcal{L}_{\text{NCE}}^{\text{opt}}$ and minimizing the loss $\mathcal{L}_{\text{NCE}}$ with back-propagation will lead to maximizing the mutual information MI between x_{t+k} and c_t.

This property is fundamental for CPC because we want, for each time step t, to maximize the quantity of useful information in c_t to be able to predict a future time step x_{t+k}.

Chapter 4

Implementations

The short-term objective of my work was to create a Python library to train and evaluate self-supervised models for speaker and language recognition. The source code is open source and available on GitHub [1]. Most of the work was dedicated to the implementation from scratch of all models presented in section 2 with TensorFlow. In addition, I had to work on modules to handle audio datasets, evaluate models on different tasks (speaker recognition, speaker verification, language recognition and data-efficiency) and create a modular system to be able to change model architectures quickly through configuration files.

However, it is noteworthy that the positive results obtained with CPC combined with NCE loss, shown in section 3.2, convinced us to focus on this technique for the continuation of our work. Moreover, CPC training objective is easier and faster to optimize as opposed to other models which could not have been trained in time before the end of the semester. Therefore, the following sections and the final experiments will be dedicated to models based on CPC.

4.1 CPC model

4.1.1 Creating the architecture with TensorFlow

CPC model is composed of three TensorFlow modules: a speech encoder to encode x_t to z_t, a recurrent neural network (RNN) to create the context vector c_t and a "predictor" to determine $\hat{z}_t$ (which is basically c_t mapped in the same dimension as z_t to compare vectors in the same latent space).

- **Speech encoder**: It consists in a simple convolutional neural network composed of 5 layers of 512 layers each. The kernel lengths are respectively 10, 8, 4, 4 and 4. The strides are respectively 5, 4, 2, 2 and 2. Each layer is followed by Batch normalization and ReLU activations. The downsampling factor of the encoder is 160, thus a signal of length 20480 produces 128 vectors z of dimension 512.
- **RNN model**: We use a standard GRU (Gated Recurrent Unit), working similarly to a LSTM, with 256 units.

[1] https://github.com/theolepage/ssl-for-slr

- **Predictor model**: It consists in a list of simple dense layers, with no activation, for each timestep we want to predict. The aim is to determine predictions with most information from c_t and to create a vector with the same dimension as the output of the speech encoder.

We implement only the forward pass of the model as TensorFlow handles backpropagation. We first encode the entire signal with the speech encoder module. During training we use signals of length 20480, resulting in 128 vectors z for a given audio signal. We keep the last 12 vectors to produce targets for the NCE loss. The first 116 vectors are fed to the RNN and then to the predictor to create 12 predictions $\hat{z}$. Finally, the NCE loss will be computed given $\hat{z}$ and the last 12 actual vectors z of the sequence.

4.1.2 Vectorization of CPC objective function

The specificity of our implementation is related to the way we designed the vectorization of CPC objective function. An extract of the source code is shown on Listing 4.1.

```python
@tf.function
def cpc_loss(nb_timesteps_to_predict, predictions, X_future_encoded):
    # Shape: (batch_size, nb_timesteps_to_predict, encoded_dim)

    batch_size = tf.shape(predictions)[0]

    losses = tf.zeros((batch_size))

    for t in range(nb_timesteps_to_predict):
        dot = tf.linalg.matmul(X_future_encoded[:, t, :],
                               predictions[:, t, :],
                               transpose_b=True)

        # Determine loss
        log_softmax_dot = tf.nn.log_softmax(dot, axis=0)
        diag = tf.linalg.tensor_diag_part(log_softmax_dot)
        losses += diag

    losses /= tf.cast(nb_timesteps_to_predict, dtype=tf.float32)

    # Compute the average loss and accuracy across all batches
    loss = tf.math.reduce_mean(losses)

    return -loss
```

Listing 4.1 – Implementation of CPC loss

To avoid iterating over each audio sample in the mini-batch, we compute for each timestep to predict, the matrix multiplication between the actual vectors to be predicted and the predictions. Then, we apply a softmax and a log operation. Thus, for each row i and column i of the resulting matrix, we have the target prediction multiplied by our prediction on the numerator and the sum of our prediction multiplied by vectors from different elements in the batch on the denominator. Thus, we only need to keep the diagonal to have the positive sample on the numerator, similarly to Eq. 2.2.

Thus, our sampling strategy for distractors (negative samples) relies on the assumption that other utterances in the mini-batch are from different speakers. It is often the case as the number of speakers is much higher than the batch size.

4.2 Improvements to CPC model

Furthermore, we introduced several improvements to CPC base model which are described in the following subsections. The objective is to give more information and more capacity to the model to reach better classification results on the final downstream task.

Some improvements are inspired by a publication that introduced changes to CPC for image recognition tasks [Hénaff et al. (2020)]. As images are two dimensional signals, we can imagine that these techniques can be successfully applied for audio data.

The complete model architecture with the new speech encoder, the two GRUs and the audio augmentation module is represented on Fig. 4.1.

4.2.1 A sinc-based speech encoder (cpc-spk-2)

In order to give more capacity to the speech encoder, we introduced a sinc-based convolutional layer which acts as a band-pass filter [Ravanelli and Bengio (2019b)]. This layer replaces the usual convolutional layer at the beginning of the encoder and has the same configuration. The intuition is that it could help the model distinguish between different groups of voice characteristics.

Moreover, we replaced Batch normalization layers by Layer normalization [Ba et al. (2016)] as some argue that it harms CPC downstream performance [Hénaff et al. (2020)].

4.2.2 Predicting frames in reverse order (cpc-spk-3)

We extend the ability of the model to predict future frames with past frames by adding a second autoregressive model which predicts the beginning of the sequence according to the following frames. In terms of implementation, we simply reverse the signal and we compute the total loss as the sum of the NCE loss for predictions in the correct and reversed order.

Thus, the principle behind CPC remains unchanged but we give more information to the model which could eventually reduce overfitting.

4.2.3 Introducing data-augmentation techniques (cpc-spk-4)

In order to learn more generalist representations, we add a speech augmentation module before our sinc-based speech encoder. The idea is to apply different transformations to the input audio signal. The transformations and their associated probability are shown on Table 4.1.

Effect	Probability	Description
Reverberation	50%	Convolve signal with a set of impulse responses.
Background noise	40%	Add noises from FreeSound and DIRHA datasets.
Frequency mask	40%	Mask some frequency band on the spectrogram.
Temporal mask	20%	Make a randomly chosen part of the audio silent.
Clipping	20%	Add saturation to the audio signal.
Overlap speech	10%	Add a different speech signal in the background.

Table 4.1 – Transformations applied to the input audio signal.

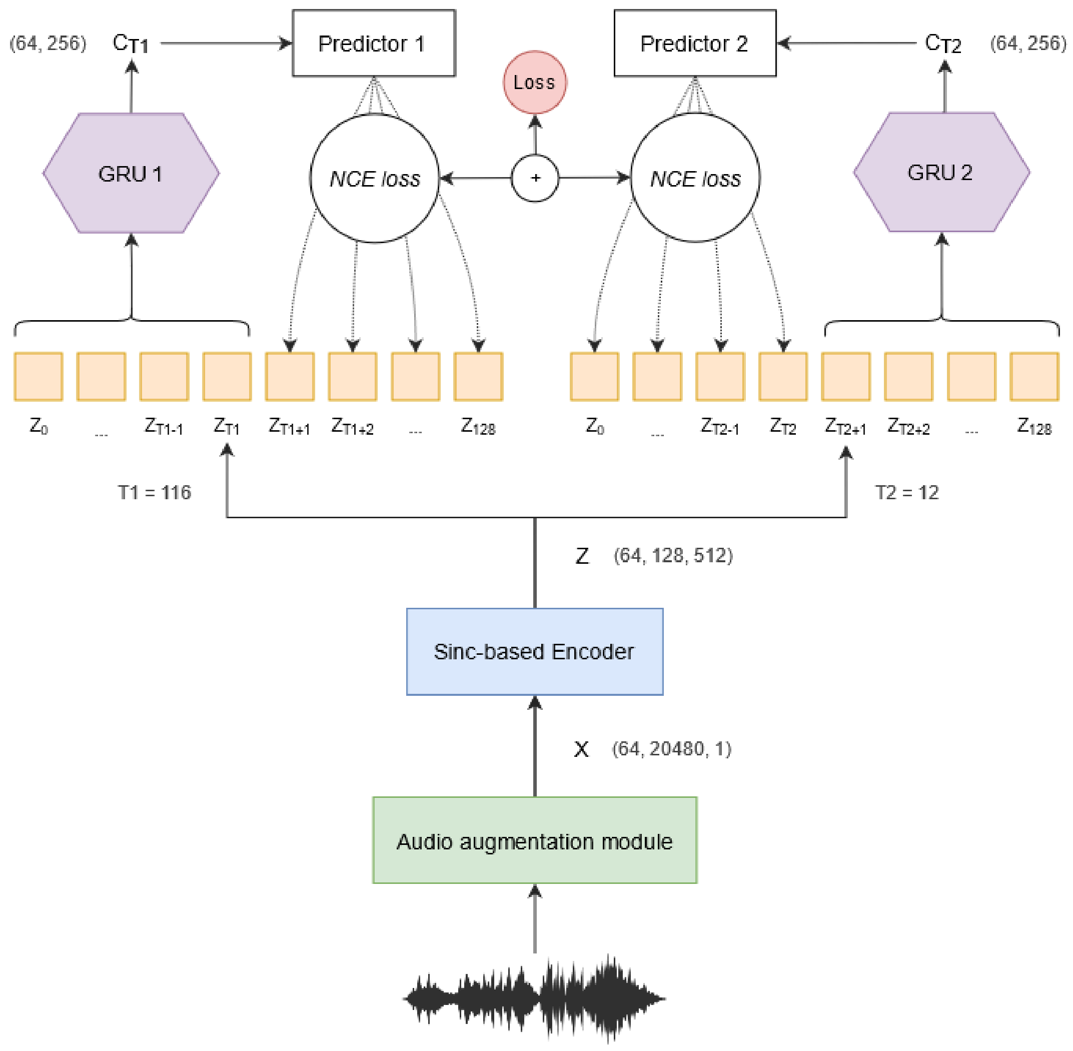

(64, 256)
C_{T1}
Predictor 1
Loss
Predictor 2
C_{T2}
(64, 256)
GRU 1
NCE loss
+
NCE loss
GRU 2
Z_0 ... Z_{T1-1} Z_{T1} Z_{T1+1} Z_{T1+2} ... Z_{128}
Z_0 ... Z_{T2-1} Z_{T2} Z_{T2+1} Z_{T2+2} ... Z_{128}
T1 = 116
T2 = 12
Z (64, 128, 512)
Sinc-based Encoder
X (64, 20480, 1)
Audio augmentation module

Figure 4.1 – CPC model architecture with our improvements.

Chapter 5

Experiments and results

5.1 Datasets

For our experiments related to speaker recognition, we considered LibriSpeech [Vassil Panayotov and Khudanpur (2015)] and its subsets *train-clean-100*, *train-clean-360*, *train-other-500* for a total of 960 hours and 2338 speakers. The data is derived from read English audiobooks from the LibriVox project. LibriSpeech was originally designed for speech recognition but it has the advantage of being carefully segmented and aligned.

Regarding language recognition, we used VoxLingua107 [Valk and Alumäe (2020)] which contains data for 107 languages with approximately 62 hours per language. It was created by automatically extracting speech segments from YouTube videos and labeled according to the language of the video title and description.

For each audio file, we choose one frame of 1.28 second which corresponds to 20480 values as both datasets are sampled at 16kHz. The validation and test sets are created by using respectively 20% and 10% of the training set.

5.2 Trainings

We rely on the model detailed in Chapter 4 by breaking it down into different configurations, described below, to conduct an ablation study.

- *cpc-spk-1*: CPC base model
- *cpc-spk-2*: Sinc-based encoder
- *cpc-spk-3*: Bidirectional GRUs
- *cpc-spk-4*: Sinc-based encoder + Data-augmentation

We first train these models in a self-supervised way (pretext task) before training a classifier on top of their representations (downstream task) as shown on Figure 5.1. We train our models with Adam optimizer [Kingma and Ba (2017)], a learning rate of 1×10^{-4}, a batch size of 64 and a L2 weight normalization factor of 1×10^{-4}. Each training stops after 5 epochs without a better validation loss and they were all performed on two NVIDIA TITAN X GPUs.

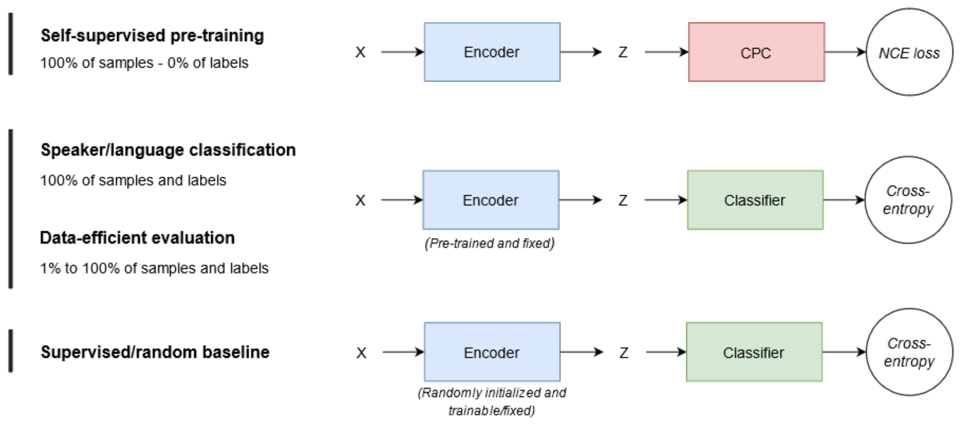

Self-supervised pre-training
100% of samples - 0% of labels
Speaker/language classification
100% of samples and labels
Data-efficient evaluation
1% to 100% of samples and labels
Supervised/random baseline
X
Encoder
Z
CPC
NCE loss
Classifier
Cross-entropy
(Pre-trained and fixed)
(Randomly initialized and trainable/fixed)

Figure 5.1 – Training and evaluation framework.

5.3 Results for speaker and language recognition tasks

5.3.1 Linear classification for speaker recognition

To assess the ability of our models to classify speakers correctly, we train a linear classifier on top of each model representations. The idea is to verify that features, learned in a self-supervised way, capture speaker identity and can be linearly separable. The classifier consists in a single linear dense layer of 256 units and a softmax layer outputting classification probabilities.

Model	Accuracy	# of params	Training time
Random initialization	1.00%	6,518M	3 hours 50 min
Supervised baseline	83.92%	6,518M	3 hours 20 min
cpc-spk-1 (Base model)	77.90%	6,518M	1 hour 30 min
cpc-spk-2 (Sinc-encoder)	72.29%	6,518M	1 hour 30 min
cpc-spk-3 (Bidirectional GRU)	86.23%	7,175M	2 hours 40 min
cpc-spk-4 (Data-augmentation)	55.10%	6,518M	1 hour 10 min

Table 5.1 – Linear classification of 2338 speakers from LibriSpeech.

We can notice on Table 5.1 that the model reaching the best accuracy is *cpc-spk-3* as it surpasses the supervised baseline. Indeed, we obtain an accuracy of 86.23% with the first and of 83.92% with the latter.

5.3.2 MLP classification for speaker recognition

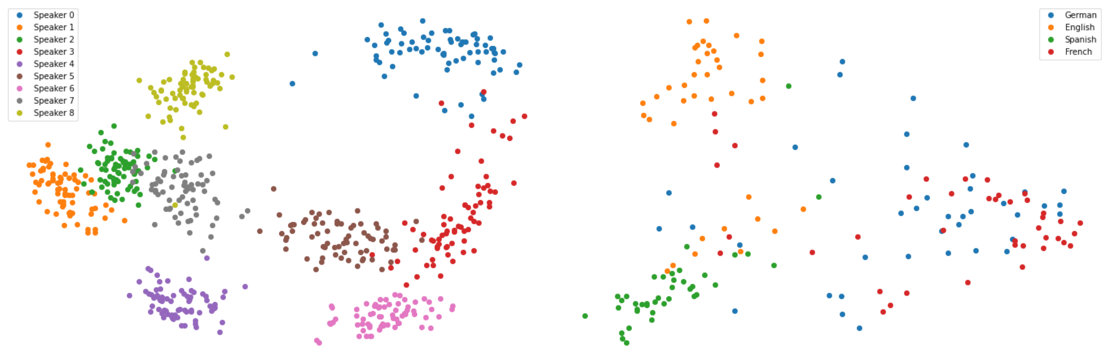
As opposed to the previous experiment, we rely on a classifier with a greater capacity as it consists in two dense layers of 512 units with ReLU activations and a final softmax layer. The objective is to evaluate the role of the classifier in previous results.

Model	Accuracy	# of params	Training time
Random initialization	2.00%	7,445M	2 hours 50 min
Supervised baseline	74.50%	7,445M	4 hours 30 min
cpc-spk-1 (Base model)	81.58%	7,445M	1 hour
cpc-spk-2 (Sinc-encoder)	79.51%	7,445M	1 hour
cpc-spk-3 (Bidirectional GRU)	87.78%	8,168M	1 hour 30 min
cpc-spk-4 (Data-augmentation)	61.09%	7,445M	1 hour 20 min

Table 5.2 – MLP classification of 2338 speakers from LibriSpeech.

As shown on Table 5.2, a multilayer perceptron allows us to reach higher accuracies, especially for *cpc-spk-3* which classifies 87.78% of speakers correctly.

5.3.3 Regarding language recognition



(a) Speaker embeddings (b) Language embeddings

Figure 5.2 – PCA on embeddings provided by the self-supervised model.

Unfortunately, we were not able to conduct experiments on the task of language recognition as the results were not convincing. The initial intuition was to use the same framework used previously but with a different dataset dedicated to language recognition. We found out that this strategy does not work well as opposed to speaker recognition.

In order to understand the cause of these bad results, we applied a PCA on the features provided by our self-supervised models trained on LibriSpeech (5.2a) and VoxLingua107 (5.2b). We can notice, that the first model is able to discern 8 speakers successfully. However, the second model struggle to distinguish audio samples spoken in German, English, Spanish and French. The underlying assumption is that language is not the most varying information among audio samples from VoxLingua107. Actually, it is easier for the model to use speaker identity to solve the contrastive task, which explains why some clusters are still visually identifiable.

Consequently, we tried to use longer context windows and reduce the number of speakers for each language in the dataset with no success. It is noteworthy that this problem represents a significant challenge as the obvious solution of having samples from only one speaker per language is not suitable nor feasible.

5.4 Data-efficient evaluation

The main purpose of self-supervised is to allow reaching supervised models accuracies while using less labeled samples. Therefore, we conducted a data-efficient evaluation on our best model (*cpc-spk-3*) applied to the task of speaker recognition. On Figure 5.3, we compare the classification accuracy of a supervised baseline with our model while varying the amount of labeled samples during training for the downstream task.

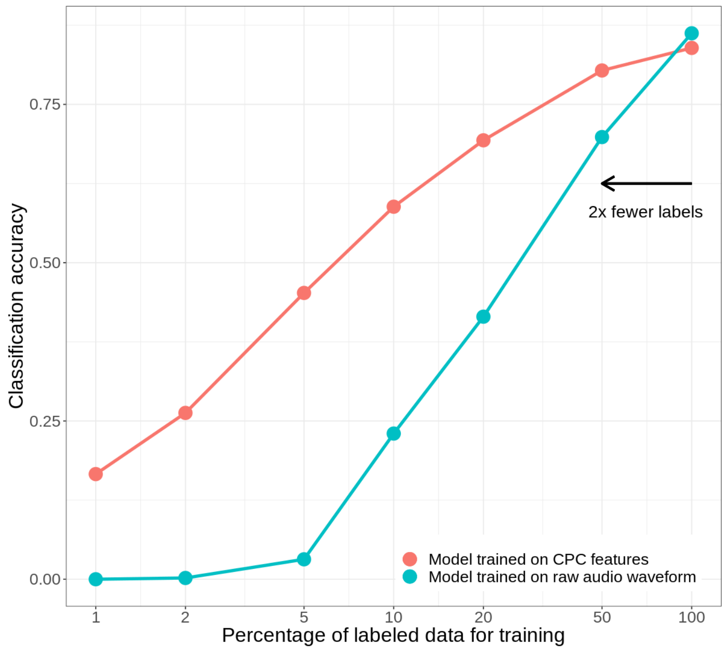


Figure 5.3 – Data-efficient evaluation of our best model on speaker classification.

It should be noted that 100%, 50%, 20%, 10%, 5%, 2% and 1% represent respectively 4200, 2100, 840, 420, 210, 84 and 42 minutes approximately for all of the 2338 speakers.

In comparison to the model trained on raw waveform, the classifier trained on our model features reaches better accuracies when reducing the number of labeled samples. Even when using half of labeled data, we get an accuracy of 80.36% surpassing the supervised baseline.

Chapter 6

Perspectives and conclusion

Self-supervised learning techniques proved to be very efficient when applied to speaker recognition. After, presenting several self-supervised from scientific literature we have shown why mutual information is a fundamental concept and how this metric can be estimated through neural networks. We were able to bring several improvements to CPC base model, introduced in [van den Oord et al. (2019)], allowing us to reach higher accuracies. Finally, we showed that the proposed framework reaches our supervised baseline accuracy while using less labeled samples.

Nonetheless, several improvements can be introduced to our work. The most important one would be to find a way to apply self-supervised learning to language recognition. Contrastive tasks is not the only unsupervised strategy and training different self-supervised modules jointly, similarly to PASE described in 2.2.4, could be a solution.

Regarding speaker recognition, our model can be significantly improved by testing different combinations of hyper parameters and changes in model architecture. Furthermore, we were not able to train all models implemented, such as PASE, vq-wav2vec and wav2vec 2.0, due to a lack of time. Finally, our experimental setup could be completed by evaluating on a larger dataset dedicated to speaker recognition (NIST SRE for instance) and by comparing our results with the few existing self-supervised models, even if they are not specifically designed for our task.

Making neural networks learn a background knowledge with no human supervision is very interesting and the transition to practice by implementing self-supervised models was very instructive.

To conclude, I believe that self-supervised learning is a promising tool and that it represents one of the key to build more intelligent speaker and language recognition systems.

Chapter 7

Bibliography

Ba, J. L., Kiros, J. R., and Hinton, G. E. (2016). Layer normalization. (page 17)

Baevski, A., Schneider, S., and Auli, M. (2020a). vq-wav2vec: Self-supervised learning of discrete speech representations. (page 8)

Baevski, A., Zhou, H., Mohamed, A., and Auli, M. (2020b). wav2vec 2.0: A framework for self-supervised learning of speech representations. (page 8)

Belghazi, M. I., Baratin, A., Rajeswar, S., Ozair, S., Bengio, Y., Courville, A., and Hjelm, R. D. (2018). Mine: Mutual information neural estimation. (page 12)

Bengio, Y., Léonard, N., and Courville, A. (2013). Estimating or propagating gradients through stochastic neurons for conditional computation. (page 9)

Gutmann, M. and arinen, A. H. (2010). Noise-contrastive estimation: A new estimation principle for unnormalized statistical models. (pages 7 and 13)

Hénaff, O. J., Srinivas, A., Fauw, J. D., Razavi, A., Doersch, C., Eslami, S. M. A., and van den Oord, A. (2020). Data-efficient image recognition with contrastive predictive coding. (page 17)

Jang, E., Gu, S., and Poole, B. (2017). Categorical reparameterization with gumbel-softmax. (page 9)

Kingma, D. P. and Ba, J. (2017). Adam: A method for stochastic optimization. (page 19)

Pascual, S., Ravanelli, M., Serrà, J., Bonafonte, A., and Bengio, Y. (2019). Learning problem-agnostic speech representations from multiple self-supervised tasks. (page 9)

Ravanelli, M. and Bengio, Y. (2019a). Learning speaker representations with mutual information. (pages 8 and 13)

Ravanelli, M. and Bengio, Y. (2019b). Speaker recognition from raw waveform with sincnet. (pages 8 and 17)

Ravanelli, M., Zhong, J., Pascual, S., Swietojanski, P., Monteiro, J., Trmal, J., and Bengio, Y. (2020). Multi-task self-supervised learning for robust speech recognition. (page 10)

Schneider, S., Baevski, A., Collobert, R., and Auli, M. (2019). wav2vec: Unsupervised pre-training for speech recognition. (page 8)

Valk, J. and Alumäe, T. (2020). Voxlingua107: a dataset for spoken language recognition. (page 19)

van den Oord, A., Li, Y., and Vinyals, O. (2019). Representation learning with contrastive predictive coding. (pages 6, 13, 14, and 23)

Vassil Panayotov, Guoguo Chen, D. P. and Khudanpur, S. (2015). Librispeech: an asr corpus based on public domain audio books. (pages 7 and 19)

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., and Polosukhin, I. (2017). Attention is all you need. (page 8)