Self-supervised learning with PASE

Ge Li
University of Montreal
Canada
ge.li@umontreal.ca

Eshwanth Baskaran
University of Montreal
Canada
eshwanth.baskaran@umontreal.ca

Balaji Balasubramanian
University of Montreal
Canada
balaji.balasubramanian@umontreal.ca

ACM Reference Format:
Ge Li, Eshwanth Baskaran, and Balaji Balasubramanian. 2022. Self-supervised learning with PASE. In *Proceedings of ACM Conference (Conference'17).* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 ABSTRACT

In this project, we implement and train a novel self-supervised learning model for speech data—Problem-Agnostic Speech Encoder (PASE) [4], a self-supervised learning system which trains an encoder model to achieve consensus among multiple workers. The downstream task used to evaluate the performance of the trained PASE model is the speaker classification. Due to limited computation resources, we use the same mini-Librispeech dataset for the encoder training and downstream task.

First, we investigate the influence of joint training on model convergence and show that joint training slows the model convergence. Second, we evaluate whether our model can learn better presentations by comparing the performance of encoder from different training epochs on downstream task. It shows that the downstream accuracy increases from 40.3% to 80.1% when the training epoch increases from 10 to 100, demonstrating the learning ability of our model. Third, we analyze the impact of different workers and data augmentation (using environment corruption) using an ablation experiment. It shows that the classification workers improve the performance while regression workers deteriorate it. The introduction of data augmentation has a negative impact on the model performance. Finally, we also implemented Q-RNN and skip connections proposed in PASE+ [6] and analyzed its impact on representation abilities. The Q-RNN alone has no significant influence on model performance, but adding both skip connections and Q-RNN increases the accuracy from 80.1% to 86.1% and makes the convergence faster.

2 INTRODUCTION

Supervised learning has achieved remarkable success in various deep learning domains, for example in computer vision [3] and speech recognition [2]. But there exists some crucial limitations related to this learning strategy. First, supervised learning models require lots of labelled data and data collection can be prohibitive or expensive (i.e. ImageNet). Second, studies have shown that they can only learn features limited to specific tasks [7].

Self-supervised learning has proved to be promising in addressing these issues in recent years. It can learn representations from the information embedded in unlabelled data, which is done by applying data transformations on input data and using the result of those transformations as labels. Therefore, a vast amount of data can be utilized while alleviating the necessity of tedious data annotations. The knowledge learned by self-supervised learning has demonstrated robust transferability in various downstream tasks.

Self-supervised learning has been successfully applied to many tasks with images and text data. It's application for speech data has been relatively unexplored. One reason for this is that speech data consists of high dimensional and variable length sequences which requires labels for effective learning. It is difficult to find a single task that can effectively deal with complex speech signals. A plausible strategy for this is to train an encoder to jointly learn from multiple learning tasks. One challenge with self-supervised learning is that there is no guarantee if the features learned by the model are useful for future tasks. The model might learn features that solve the given task well but would not help it understand the speech data and derive useful features from it. This challenge can be solved by forcing the neural network model to perform well in multiple self-supervised learning tasks. Each task imposes a different constraint on the model and the model is forced to learn features that are more general and transferable. In this project, we implemented a model of such characteristics—the Problem-Agnostic Speech Encoder (PASE) [4].

The PASE model has two components, one encoder followed by multiple cooperative workers. The encoder tries to learn useful features from speech data and the subsequent cooperative workers use the features learned by the encoder to perform multiple tasks. The encoder consists of several layers so that they can learn complex features from the data and the workers have a relatively small capacity. This design forces the encoder to learn robust and universal representations in order to achieve consensus across different tasks.

The experiments find that the PASE model can learn complex features directly from the raw waveform. This useful property enables us to build good representations from massive amounts of unlabelled speech data. Another potential application is that the PASE encoder can be used as a pre-trained network with general purpose features that can be used for a variety of downstream tasks such as speaker identification and automatic speech recognition.

3 PROPOSED APPROACHES

In this project, we follow the strategies proposed in [4], where the authors developed an architecture the problem-agnostic speech encoder (PASE) (Figure 1), which can learn robust, general and

Conference'17, July 2017, Washington, DC, USA

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

transferable speech representations. The encoder in the model is

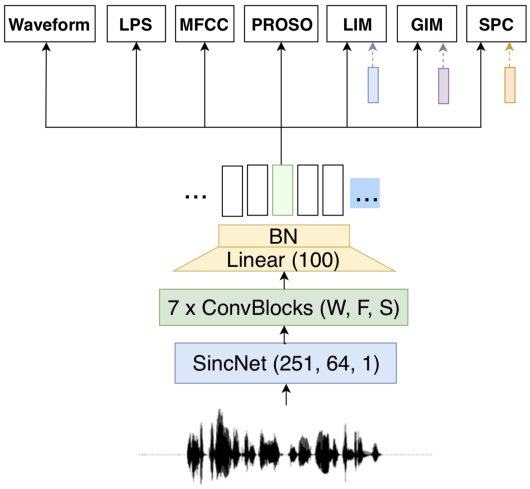

Figure 1: The PASE architecture from [4].

followed by seven different multi-layer perceptron (MLP) workers. The workers share the same representations learned by the encoder and perform different and independent tasks based on the learned representations. The encoder and workers are trained cooperatively by minimizing the average loss of different workers.

3.1 Encoder

The architecture of the encoder is illustrated in Figure 1. The first layer is the SincNet model [5], a novel Convolutional Neural Network that learns band-pass filters based on parametrized sinc functions. The SincNet layer has a kernel width of $W = 251$, filter number $F = 64$ and a stride $S = 1$. The SincNet layer is followed by 7 convolutional blocks, and each block is composed of a 1-D convolution layer followed by batch normalization (BN), and a multi-parametric rectified linear unit (PReLU) activation unit. For these 7 convolutional blocks, the parameters used are: kernel widths $W = \{20, 11, 11, 11, 11, 11, 11\}$, filter numbers $F = \{64, 128, 128, 256, 256, 512, 512\}$, and strides $S = \{10, 2, 1, 2, 1, 2, 2\}$. The subsequent layer is a linear transformation which projects 512 features to embeddings of dimension $d = 100$. These embeddings are further transformed by a non-affine BN layer into the final representations.

Effectively given an input waveform on T samples, the number of output feature vectors (frames) is $N = \frac{T}{160}$ and each vector has a dimension of $d = 100$. When the sampling rate $S_r = 16000$, this operation is equivalent to a 10ms stride and the receptive field is about 150ms.

3.2 Workers

The workers implemented in this project consists of 4 regression workers (Waveform decoder, LPS, MFCC and PROSO) and 3 binary discriminator workers (LIM, GIM and SPC). The workers are simple and shallow in comparison to the encoder because we want to force the encoder to do all the learning.

We will first introduce regression workers.

- **Waveform worker (decoder):** This worker is trained as an auto-encoder to reconstruct the input waveform signal. The PASE representaion is upsampled by a factor of 160 by three transposed convolutional blocks. The parameters for these three blocks are: kernel sizes $W = \{30, 30, 30\}$, filter numbers $F = \{512, 256, 128\}$ and strides $S = \{4, 4, 10\}$. Then an MLP layer of 256 PReLu units is used with a single output unit per frame. The loss function is the mean absolute error (L_1).
- **Log power spectrum (LPS):** This worker is trained to predict the LPS of input waveform signal using an MLP layer of 256 PReLu units. The ground truth LPS of the signal is calculated using a Hamming window of 25 ms, a step size of 10 ms and the number of fft points of 2048. The loss function is the mean squared error (MSE).
- **Mel-frequency cepstral coefficients (MFCC):** This worker is trained to predict the MFCC of input waveform signal using an MLP layer of 256 PReLu units. 20 coefficients are extracted from 40 mel filter banks. The loss function is the mean squared error (MSE).
- **Prosody:** This worker is trained to predict the prosody of input waveform signal using an MLP layer of 256 PReLu units. In this study, the prosody feature consists of four basic features: the interpolated logarithm of the fundamental frequency (lf_0), voiced/unvoiced probability (uv), zero-crossing rate (zcr), and energy. lf_0 and uv are calculated using *pysptk* while zcr and energy are calculated using *librosa*. We use a window length of 20 ms and step size of 10 ms. The minimum and maximum fundamental frequency are 60 Hz and 300 Hz, respectively. The loss function is the mean squared error (MSE).

Then we introduce the binary discriminator workers. Each discriminator work depend on its own sampling strategy that draw an anchor x_a, a positive sample x_p and a negative sample x_n from the pool of training set PASE representations. An MLP is used to minimize the following loss:

$$L = \mathbb{E}_{X_p}\left[\log\left(g\left(x_a, x_p\right)\right)\right] + \mathbb{E}_{X_n}\left[\log\left(1 - g\left(x_a, x_n\right)\right)\right],$$

where g is the discriminator function, and $\mathbb{E}_{X_p}$ and $\mathbb{E}_{X_n}$ are expectations over positive and negative samples, respectively.

- **Sequence predicting coding (SPC):** x_a is the encoded representation of a single random frame, while x_p is the representation of randomly extracted, 5 consecutive future frames and x_n is the representation of randomly extracted, 5 consecutive past frames. We avoid sampling inside the current-frame receptive field (150 ms) and we sample up to 500 ms away from x_a. Its sampling strategy is illustrated in Figure 2.
- **Local info max (LIM):** x_a is the encoded representation of a waveform chunk from a random sentence, x_p is the encoded representation of a waveform chunk from the same sentence to the anchor and x_n is the encoded representation of a waveform chunk from a random sentence that belongs

to a different speaker. This worker can lean a representation embedding speaker identify information. Its sampling strategy is illustrated in Figure 3.

- **Global info max (GIM):** x_a is the averaged encoded representation of a waveform chunk from a random sentence across different frames. x_p is similarly derived from a waveform chunk within the same sentence to the anchor and x_n is derived from a random sentence that belongs to a different speaker. Effectively, the samples of GIM is obtained by taking average across the frame axis of LIM sample. GIM workers can learn global representations complementary to the local representations learnt by LIM.

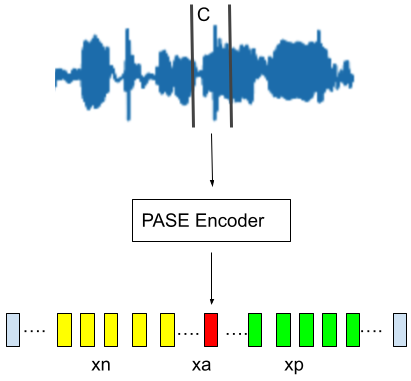

Figure 2: The sampling strategy of SPC worker.

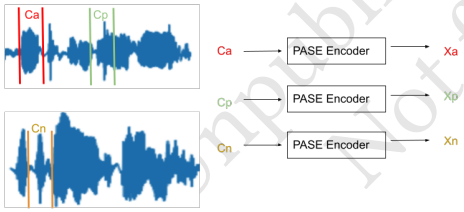

Figure 3: The sampling strategy of LIM worker.

3.3 Downstream Classifier

To evaluate if the model learns good representations of data, we perform a downstream speaker identification task by adding a classification layer on top of the trained encoder. The classification layer is an MLP layer with 100 neurons with Leaky Relu activation. The output layer consists of 28 neurons with softmax activation.

4 EXPERIMENTAL SETUP

In this section we will discuss the dataset and hyperparameters settings.

4.1 Dataset

The dataset we used for the experiments is mini-Librispeech. Mini-Librispeech dataset is a small subset of the Librispeech dataset created by Librivox projects and contains speech from many audiobooks. This small dataset consists of 5 hours of English speech at a sampling rate of 16 KHz spoken by 28 speakers. We used this dataset for both the training of the PASE model and the downstream task of speaker classification.

4.2 Other Hyperparameters

- **Input Data Corruption:** We conduct experiments using both clean data and data corrupted with environment noises. OpenRIR is used to corrupt the data by adding noise and reverberation.
- **other experiment settings for PASE model:** To train the PASE model, we used waveforms with a sampling rate of 16000, batch size of 32, learning rate of 0.0005 for both the encoder and workers with a decay factor of 0.5 every 20 epochs and we trained the model for a total of 200 epochs. Before training the model end-to-end, we conduct train each worker individually to make sure every worker is capable of learning from the dataset with the given model architecture. We trained the model initially for a total of 8 configurations (Table 1). Note that the 6 of those configurations have one worker removed and they have been trained without input data corruption. Only one of the experiment has input data corruption with noise. Later, we tried to add Q-RNN and skip connections to the encoder network and Table 2 shows the performance of the model on downstream task.
- **other experiment settings for downstream task:** For training the downstream task, we used a sampling rate of 16000 and a batch size of 32. The learning rate was 0.001 at the start of the training and it was decreased gradually over the course of the 50 epochs to 0.0001. The encoder model at epochs 10, 50, 100, 150 and 200 were checkpointed and used in the downstream task. We can infer from the results table that the PASE model checkpointed at 100^{th} epochs performed better than the one checkpointed in the 10^{th} epoch. This clearly shows that the PASE model is able to learn good representations of data with epochs.

5 EXPERIMENTAL RESULTS

5.1 Training PASE Encoder

To test if the workers are implemented properly, we first train each worker individually for 75 epochs (or 2850 iterations). The results show that the losses of all workers decreases over training time, manifesting that our model is working properly. With this being established, we then train all workers jointly for 200 epochs (or 7600 iterations). Figure 4 shows that training multiple workers converges slower than training each worker individually. In addition, the final losses of individually trained workers are smaller. This is expected since joint training is considered to be more challenging as it requires consensus among all workers. Two exceptions are the waveform decoder and the MFCC worker, which shows that the losses of joint training converges at a similar speed to the

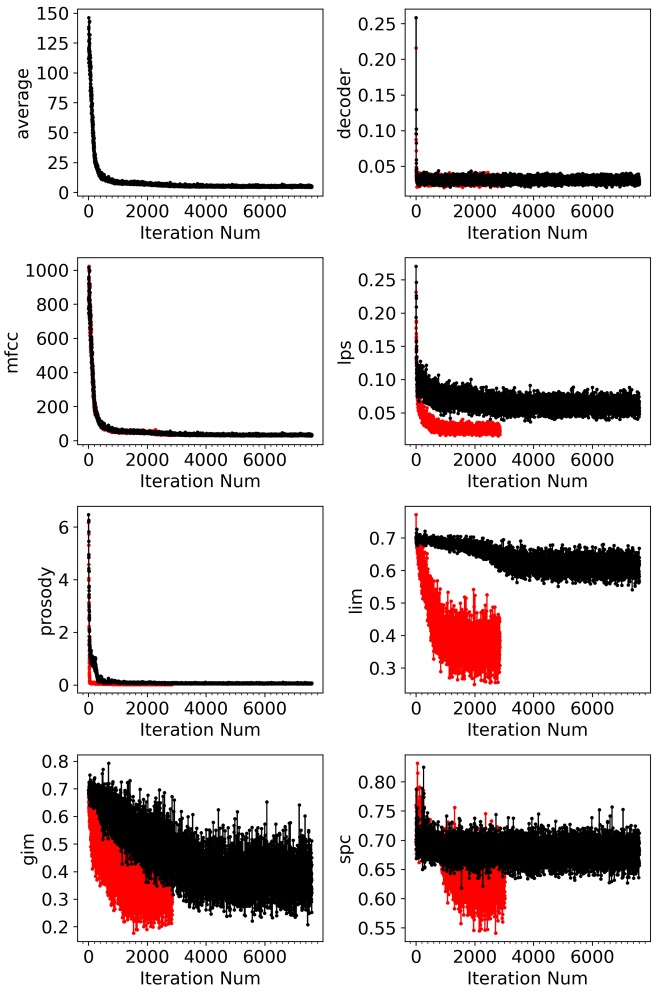

Figure 4: Losses of different workers as functions of training time. Black curves: losses when all workers are trained collectively; red curves: losses when each worker is trained individually. 1 epoch = 38 iterations.

individual training scenario. For the MFCC worker, this may be due to that the magnitude of its loss is more than 3 magnitudes larger than other workers, whether it is trained individually or jointly has no influence on its convergence. However, the behavior of the waveform decoder worker still lacks an explanation.

5.2 Downstream experiments

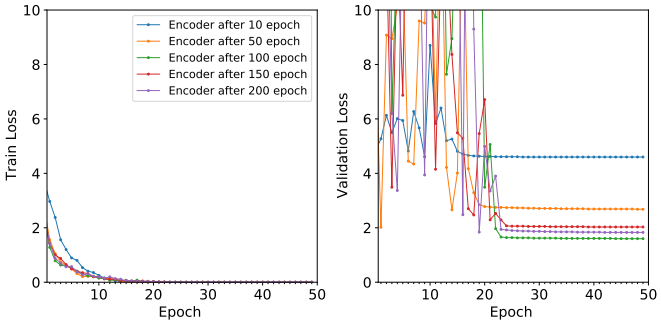

Figure 5: Train and validation loss for downstream task (Speaker identification) when incorporated with learned representations from PASE encoders trained after different epochs.

Table 1: Validation accuracy for each configuration for various epochs of PASE model for the downstream task of Speaker Identification.

Model	E 10	E 50	E 100	E 150	E 200
All workers w/o noise	40.3	66.2	**80.1**	76.2	72.8
- Waveform	68.2	**83.4**	78.1	78.1	82.1
- MFCC	74.8	83.4	90.07	**90.73**	82.8
- LPS	45.7	80.8	76.2	**82.1**	78.8
- Prosody	60.3	**81.5**	76.2	78.8	80.8
- LIM	55.6	68.2	**78.8**	78.1	78.8
- GIM	57.6	75.5	**76.8**	74.8	76.2
- SPC	51.7	74.8	**80.8**	79.5	76.8
All workers w/ noise	38.4	61.8	68.2	64.9	**69.5**

In Table 1, it can be seen that the PASE encoder models trained for 50 epochs and above perform much better than PASE encoder model trained for 10 epochs for all the downstream experiment configurations. This clearly shows that the PASE model learns useful features that can be used for downstream task of speaker identification.

We have two unexpected results. The first unexpected result is that the PASE encoder model without the MFCC worker has the best performance, better than the PASE model with all workers.The second unexpected result is that in many configurations, the PASE encoder trained for 50, 100 and 150 epochs show the best validation accuracy and not the PASE model trained for 200 epochs. The first unexpected result could be due to that the magnitude of the MFCC loss is at least 3 times larger than other losses. This may prohibit the model to extract information from other workers. The second unexpected results could be due to the small dataset size, therefore the model can easily overfit the given tasks instead of learning a robust and universal representation.

Table 2: Validation accuracy on the downstream task on adding QRNN and skip connections incrementally.

Model (clean data)	E 50	E 100	E 150	E 200
All workers	66.2	**80.1**	76.2	72.8
+ QRNN	64.2	77.5	78.8	**80.1**
+ skip connections	76.7	80.8	85.4	**86.1**

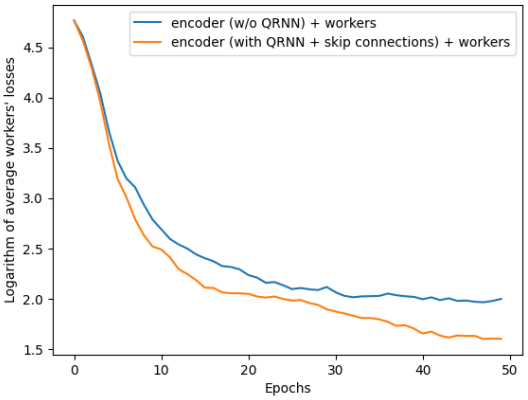

Figure 6: Impact on average train loss when Q-RNN and skip connections are added

- **Q-RNN:** QRNN, or Quasi-Recurrent Neural Network [1], is a type of recurrent neural network that alternates convolutional layers, which applies in parallel across time steps, and a minimalist recurrent pooling function that applies in parallel across channels. Q-RNN was developed by aggregating the advantages of CNNs (process variable sized input in parallel) and RNNs (handle recurrent data). In short, Q-RNN is equivalent to an LSTM but can be trained across time steps in parallel using masked convolutions. Q-RNNs are useful in that it can model long-term dependencies in the given input signal i.e. it can capture dependencies of features in the earlier part of the sequence to the features in the latter part of the sequence. This helps to understand the structure of the input signal in much more detail.

From Table 2, we can see that QRNN and skip connections are helpful in learning better representations and hence, a better performance on the downstream task. The Q-RNN helps to model long-term dependencies in the input and hence, the encoder model can understand the underlying structure of the input signal in a better way. We can infer this from the increase in accuracy on the downstream task with epochs. But from 150^{th} epoch to 200^{th} epoch, the increase in validation accuracy is small. This could be due to that the encoder has already learned sufficient information from the small dataset (miniLibrispeech) by 150^{th} epoch.

Skip connections transfer extracted feature information from every convolution layer to the final layer of the encoder network. This means that the features collected at various levels starting from simple features in the first layer to more complex features in the final layer are appended together. Therefore, we suggest that the final representation of the input signal from the encoder is feature-rich and thereby, these feature-rich representations greatly improve the performance of the downstream task.

These discussions can be inferred from the average train loss curves in Figure 6. It can be seen that the model with Q-RNN and skip connections converged quicker than the case without Q-RNN. This intuitively makes sense since the model is learning the underlying representation better with Q-RNN and skip connections at every epoch and hence, the quick convergence.

6 FUTURE WORK

In the limited time we had, we develop the self-supervised system with we add components incrementally to increase the performance of the system. But due to time and resource constraints, we were only able to train the self-supervised model and the downstream task on a subset of the actual Librispeech dataset. As part of the future work, we would be training the system on actual Librispeech dataset to obtain a robust self-supervised encoder model. This is expected to give better results. Also, we would be working on adding several downstream tasks to get the performance of the self-supervised system in various scenarios and thereby a better estimate of its performance.

In addition, we would like to explore the usage of transformers as part of the self-supervised system and we think that could also help to yield a robust and generalized self-supervised model.

7 CONCLUSION

We implemented an effective and data efficient self-supervised learning model, PASE. joint training with multiple workers tend to slow down the convergence compared to individual training. Our experiment suggests that our model can learn better representations with increasing training epoch and the features learned is transferable to downstream Speaker Identification task. We also show that the implementation of both Q-RNN and skip connections can improve model performance. In addition, we also obtain many unexpected results. First, the ablation experiment shows that the inclusion of regression worker damages model performance while the classification workers are beneficial. Second, we show that the representations learned by the encoder gets worse when the training epoch keeps increasing. This seems to suggest that the optimal training epoch can vary for different experiment configurations. Further study is required to confirm if this observation is valid.

8 STATEMENT OF CONTRIBUTIONS

We hereby state the all the work presented in this report is that of the authors.

G. Li participated in coding the recipe, running experiments on training the PASE encoder and ablation experiments, writing the report, making figures and discussion. In particular for the coding part, G. Li built the first prototype of working recipes under Colab (it includes train.py, train.yaml, the PASEEncoder and three workers (*LPS*, *MFCC* and *Prosody*)), implemented 6 workers and corresponding labellers (*LPS*, *MFCC*, *Prosody*, *GIM*, *LIM* and *SPC*).

For experiments, G. Li also performed experiments for worker debugging and testing the influence of joint training on convergence rate.

Balaji Balasubramanian participated in coding the recipe, running the experiments, writing the report, and discussion. In particular for the coding part, Balaji built the prototype of working recipes under Colab, collaborated and implemented the Speaker Identification Downstream task , partially implemented the Asr downstream task for the recipe, and programmed 2 workers Lps and Lim for the recipe. On the experimental side, Balaji performed downstream experiments, and trained 2 Pase encoder-workers models. On the report side, Balaji wrote the introduction and experimental setup and a part of abstract, experimental result and conclusion.

Eshwanth Baskaran participated in coding the recipe, running the experiment, writing the report and discussion. For the coding part, Eshwanth worked on structuring the recipe initially, making the recipe compatible to train multiple models end-to-end in speechbrain and designed the code to add new workers easily. He also implemented the waveform-decoder worker and its labeller, and then went on with collaborating and implementing the speaker classification downstream task using minilibrispeech dataset. Finally, he coded and added the QRNN and skip connections modules to the encoder network of PASE along with documenting the entire work. On the experimental side, Eshwanth performed experiments to debug decoder worker (using overfitting experiments) and also analysed the rate of learning of every worker when it was trained individually vs the case with all workers. Finally, he performed experiments running the PASE system with QRNN + skip connections and analysed its impact.

REFERENCES

[1] James Bradbury, Stephen Merity, Caiming Xiong, and Richard Socher. 2016. Quasi-recurrent neural networks. *arXiv preprint arXiv:1611.01576* (2016).

[2] Geoffrey Hinton, Li Deng, Dong Yu, George E Dahl, Abdel-rahman Mohamed, Navdeep Jaitly, Andrew Senior, Vincent Vanhoucke, Patrick Nguyen, Tara N Sainath, et al. 2012. Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups. *IEEE Signal processing magazine* 29, 6 (2012), 82–97.

[3] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. 2012. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems* 25 (2012), 1097–1105.

[4] Santiago Pascual, Mirco Ravanelli, Joan Serra, Antonio Bonafonte, and Yoshua Bengio. 2019. Learning problem-agnostic speech representations from multiple self-supervised tasks. *arXiv preprint arXiv:1904.03416* (2019).

[5] Mirco Ravanelli and Yoshua Bengio. 2018. Speaker recognition from raw waveform with sincnet. In *2018 IEEE Spoken Language Technology Workshop (SLT)*. IEEE, 1021–1028.

[6] Mirco Ravanelli, Jianyuan Zhong, Santiago Pascual, Pawel Swietojanski, Joao Monteiro, Jan Trmal, and Yoshua Bengio. 2020. Multi-task self-supervised learning for robust speech recognition. In *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 6989–6993.

[7] Michael T Rosenstein, Zvika Marx, Leslie Pack Kaelbling, and Thomas G Dietterich. 2005. To transfer or not to transfer. In *NIPS 2005 workshop on transfer learning*, Vol. 898. 1–4.