Project 1: *Self-supervised pretraining for phoneme recognition, and generalization on foreign languages*

Project report: *Algorithms for speech and natural language processing (MVA 2022)*

Clément Apavou, Younes Belkada, Léo Tronchon, Arthur Zucker
MVA Master's program
École Normale Supérieure Paris-Saclay
firstname.lastname@ens-paris-saclay.fr

1. Introduction

The scarcity of annotated data, and the heavy cost of producing them, limits our ability to train deep neural network for audio processing tasks. Therefore, the speech community developed feature learning methods with a minimal need for annotated data, which mostly fall under unsupervised and self-supervised techniques.
Recently, the rise of self-supervised learning methods for textual modality [8] has outperformed state-of-the-art methods on downstream tasks, by fine-tuning the pretrained models on a relatively small amount of data. These approaches have recently been tested for other modalities such as images [5, 19] and audios [11, 16, 2, 7].
Phoneme recognition is an exciting challenge that involves processing a raw audio recording and predict the corresponding sequence of phonemes that are pronounced by the speaker. Throughout this project, we will compare specifically three different self-supervised models, Wav2vec [16, 2] (2019, 2020), HuBERT [11] (2021) and WavLM [7] (2022) pretrained on a corpus of English speech that we will use in various ways to perform phoneme recognition for different languages with a network trained with Connectionist Temporal Classification (CTC) algorithm [10]. Different questions will be addressed:

- *What is the impact of choosing English as a pretrained language, especially for languages that are very different from English? Which method(s) works best for transferring knowledge from English to other languages?*
- *What is the influence of the abundance of training data on the performance of models?*
- *Which method allows to extract the best features for phoneme recognition?*

We will address these questions by drawing conclusions from our experiments.

2. Materials & Methods

2.1. Common Voice datasets

We used the Common Voice [1] dataset which is a speech corpus containing many languages from different language families, powered by the voices of volunteer contributors around the world. We used in particular the common voice dataset from Hugging Face dataset hub, which provides a train/validation/test split as well as useful processing tools and functions. The dataset comes with the ground truth annotation of the text transcript that was read by the speaker. We had to convert the original transcript to a sequence of phonemes in order to obtain the phoneme transcription. We will cover this aspect in the next section.

2.2. Converting transcripts to sequences of phonemes and tokenization

In order to convert the target sentences into sequences of phonemes we used the Phonemizer [3] library combined with the espeak-ng backend that supports over 100 languages. This is part of a pre-processing, which phonemizes the entire dataset, and is performed for every language we studied. Then, in order to compute the loss, we need to tokenize each phoneme, for this we used Hugging Face's phoneme tokenizer Wav2Vec2PhonemeCTCTokenizer which allows the encoding and decoding of the targets and the predictions into tokens. *(We set the* `do_phonemize` *argument to* `False` *as we realized that it was not optimized).*
Hugging Face's tokenizer requires a dictionary listing all the phonemes of a language which differs according to the language. At first, we used the PHOIBLE database [13] which lists every phoneme from a large number of languages (over 2000), but we realized that the *phoible* set of characters were different from the ones used by the phonemizer. Thus the tokenization was wrong, and many $<$ unk $>$ token appeared in our targets, which is why we

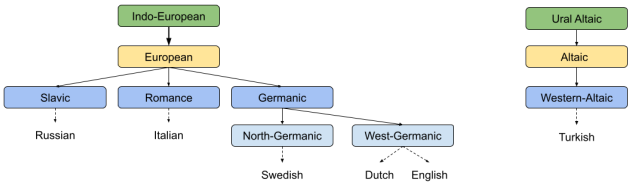

Figure 1. Representation of the language family tree of the 5 studied languages

Language	number of phonemes
Swedish	40
Turkish	47
Russian	49
Dutch	52
Italian	60

Table 1. Number of phonemes of the 5 studied languages. We add 5 special tokens in addition to these phonemes: < s >, < /s >, < unk >, < pad > and |.

used the works of [15] that provides phonemes dictionaries for 10 languages. We can group these different languages by family:

- *Romance*: Spanish, French and Italian
- *East Slavic*: Russian
- *West Germanic*: Dutch
- *North Germanic*: Swedish
- *Turkic*: Turkish, Tatar and Kirghize
- *Sino-Tibetan*: Mandarin (Taiwan)

A simplified language family tree can be found in 1. We chose to study one language from each group. Unfortunately, we did not manage to use the *Mandarin* dataset because of phoneme compatibility problems between the phonemizer, the phoneme dictionary and the espeak-ng backend. The number of phonemes of each language is reported in table 1.

2.3. Pretrained models

The aim of pre-training large models through self-supervision is to learn how to extract meaningful features from a raw set of data. The representations learned from the pretrained model would be powerful and meaningful enough to outperform models trained from scratch, sometimes with smaller amount of data. The three approaches we studied, Wav2vec2, WavLM and HuBERT are briefly described in appendix 5.2.

We used models hosted by Hugging Face [9], that were pretrained on 960 hours of **English** audio data from Librispeech [14] dataset on 16kHz sampled speech audio. Except for WavLM *Large* which was pretrained on MIX-94K (60K hours of Libri-Light [12], 10K hours of GigaSpeech [6] and 24K hours of VoxPopuli [18]). The following pretrained models were used:

- Wav2vec2 *Base*: `facebook/wav2vec2-base-960h`
- WavLM *Base*: `microsoft/wavlm-base`
- WavLM *Large*: `microsoft/wavlm-large`
- HuBERT *Large*: `facebook/hubert-large-ls960-ft`

The CNN encoders of every model extracts features of size 512. However, the pretrained HuBERT is larger as its transformer processes features of size 1024 against 768 for Wav2vec2 and WavLM. Moreover, it is twice as deep, 24 layers against 12 for the two others. In order to properly compare the feature extraction capacity of the pretrained models, it would have been better to have transformers of the same size but unfortunately there was no pretrained *base* HuBERT model available on Hugging Face (the pretrained weights are from the *Large* architecture and not the *Base*). As training large models requires more time, we decided to first train Wav2Vec2 and WavLM *Base*, and then compare HuBERT and WavLM *Large*.

2.4. Network architecture using CTC

The overall network architecture of the 3 models share some common design. It is described in figure 2. It is made of a CNN encoder, a feature projector, a transformer encoder and a linear layer used as the language modeling head. This last layer is not pretrained, and leverages the extracted features from pretrained models to perform phoneme recognition with the CTC algorithm. The output of the linear layer is the size of the vocabulary which is the number of phoneme of the language plus the special tokens. We used models from the Hugging Face library : Wav2Vec2ForCTC, WavLMForCTC and HubertForCTC. The models are initialized using the weight files mentioned in 2.3. For the loss function, we used the Connectionist Temporal Classification loss (CTCLoss) from PyTorch.

2.5. Evaluation

We evaluated our models using the Phoneme Error Rate (PER), as it is a common metric to evaluate phoneme recognition. The lower the PER the better the model is. We also compared the behavior of the loss function and the PER on the validation set.

2.6. Training Code

Our Github repository contains our implementation for this project. We built from scratch a very clean and easy-to-use training code in PyTorch Lightning. Everything is automated from the download of the common voice database to the creation of the labels by the phonemizer, and the retrieval of phoneme dictionaries obtained from [15] to be

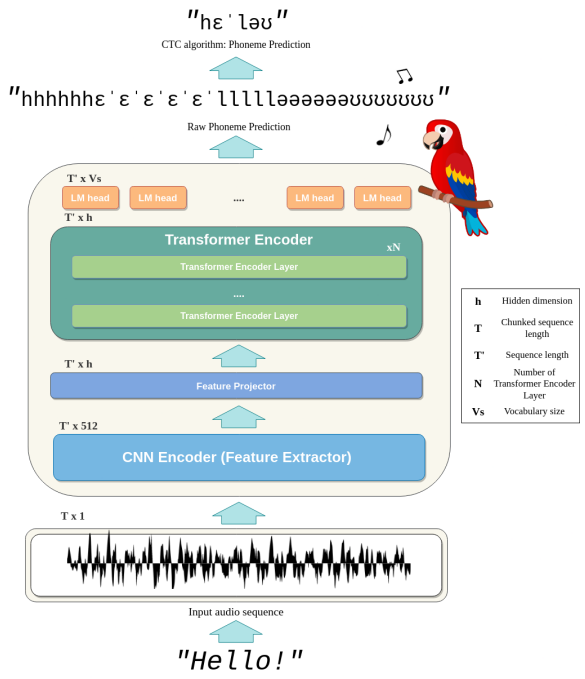

Figure 2. Diagram of the models used for the experiments. $N = 22$ and $h = 1024$ for HuBERT *Large* and WavLM *Large*, and $N = 11$ and $h = 768$ for Wav2vec2 *Base* and WavLM *Base*.

used by the tokenizer. A pretrained model (Wav2Vec2, HuBERT or WavLM) can be easily chosen using the command line, and the same goes for the language of the dataset on which the phoneme recognition must be trained.

3. Experiences, Results & Observations

3.1. Audio pre-processing

We applied various pre-processing to facilitate the training of the various models. First, we re-sampled audios to match the sampling rate of the pretrained dataset (16kHz) and normalized them. Then, we removed blanks from the training data using the librosa python library and removed large audios with a duration greater than 5 seconds to prevent overloading our GPUs' memory. For each batch, which are created on-the-fly, the audios are padded to the same length. We only used a batch size of 2 but we applied an accumulation of gradients by a factor of 16 batch during the optimization which is equivalent to using a batch size of 32.

3.2. Experiments

We focused mainly on three experiments, fine-tuning, freezing the backbone and measuring the impact of varying the amount of training data when the backbone is freezed. The last experiment focused only on the Swedish language.

3.3. Training setup

As detailed in the section (2.2), we chose to train each model on one of the 5 family: *Italian*, *Russian*, *Dutch*, *Turkish* and *Swedish*, with each having a different amount of data. We trained our models until convergence, relying on the validation PER. We used the AdamW optimizer, with a learning rate adjusted according to the experience. Trainings were launched on an NVIDIA GeForce RTX 3080, Google Colab Pro+ GPUs (Tesla V100-SXM2) and on Google Colab Free (Tesla K80) GPUs. At the end of each training, we evaluated the quality of our model on the test set.

3.4. Experiments tracking

We used the Weights & Biases (wandb) [4] library to track our experiments. For each run, we logged the configuration file with all the hyper-parameters and the global seed to be able to reproduce the results if necessary. The validation and train loss and PER as well as sample audio files with the sentences, the target phonemes and the predictions of the model can be visualized. This allows us to evaluate both quantitatively and qualitatively our models during training. All experiments are available on the following link.

3.5. Results & Observations

We report our main results in tables 3, 4 and 5. The number of hours of training data indicated in the tables were computed by multiplying the number of audio files in the training set by 5 (in seconds, since we take only audio file smaller than 5sec) and reported in hour. This is an upper bound since every audio does not necessarily lasts than 5 seconds. To interpret results from table 3 and 4, we must take into account that the performance of models on different languages is related to their proximity to the pretrained language, (which in our case is English) and the amount of training data. In table 2, we detailed the genetic proximity between English and the 5 other languages, as well as their language family. English is part of the *West Germanic* family.

3.5.1 Fine-tuning vs Frozen Features

Fine-tuning Overall, the models perform well when fine-tuned on the phoneme recognition task, both quantitatively and qualitatively (table 3). Performances appears to vary with the number of hours of available training data (e.g Italian) and the proximity of the language to English (e.g Dutch). Indeed, Dutch gives the best PER for every methods despite that it was trained on less data than some languages because it is closer to English since they are part of the same language family.

Language	Language Family	Proximity with English
Swedish	*North Germanic*	26.7
Dutch	*West Germanic*	27.2
Italian	*Romance*	47.8
Russian	*Est Slavic*	60.3
Turkish	*Turkic*	92.0

Table 2. Genetic proximity between languages studied and english computed here. [1, 30]: Highly related languages, [30, 50]: Related languages, [50, 70]: Remotely related languages, [70, 78]: Very remotely related languages, [78, 100]: No recognizable relationship.

One notable exception to the general trend is Turkish. Surprisingly, Turkish has a very low amount of training data and is the farthest language when compared to English, but it is also the second best performing language when we fine tune the models. Unfortunately, we couldn't find a sensible explanation for this phenomenon, and this discrepancy disappears when using frozen features.

Frozen features When we use frozen features, as expected the best PER is higher (table 4). This is due to the fact that the feature extractor uses only the features it has learned to extract on a corpus of English speech during his pretraining it does not specialize its feature extraction for the phoneme recognition task and for the target language. Languages with a good genetic similarity with English thus tend to give better results in proportion to the number of hours of training data used, e.g, the 3 different pretrained methods perform better on Dutch rather than Russian (which is further away from English) despite having less training data. However, the results on Swedish and Russian are pretty similar, as the proximity of Swedish to English is balanced out by the very low amount of training data.

3.5.2 Wav2vec2 vs WavLM vs HuBERT

Which method extracts the best and most relevant features for phoneme recognition? Note that it doesn't make much sense to compare *Large* models with *Base* models, so first, lets compare Wav2vec2 *Base* and WavLM *Base*. According to the results of the table 4, WavLM *Base* gives much better results than Wav2vec2 *Base* for all languages with a difference in test PER, which is between $\sim 11\%$ and $\sim 6\%$, we deduce that WavLM seems to be better than Wav2vec2. Now, let's compare WavLM and HuBERT. WavLM *Large* gives better results than HuBERT *Large* for most of the languages, and on average is better (28.31% vs 30.75% PER on the test set). We conclude that WavLM extracts better features than Wav2vec2 and HuBERT for the phoneme recognition task.

3.5.3 Importance of the amount of training data

As explained in the section 3.5.1, the number of hours of training data greatly influences the performances of the model. We measured its impact on the the Swedish language using frozen features. The results are available in table 5 and represented in figures 5.4 and 5.4. We observed that the more data we use, the better the results are (as expected). For HuBERT, with 10 min of speech data, we get a PER of 39.38% on the test set. This is improved by using 3 hours of training data as we manage to reach 32.68 % PER. However, as can be seen in table 5.4 and 5.4, as we increase the amount of training data, it translates less and less to an increase in performance. This is line to what is usually observed when training.

3.5.4 Comparing our results with the literature

In order to evaluate the consistency of our results in relation to the literature, we have identified the recent work from Rivière et al. [15] that attempts to learn meaningful representation of phonemes through Contrastive Predictive Coding. The authors pretrained their model on the LibriSpeech dataset and fine-tuned a small linear head on top of the frozen features. We can compare roughly our results from table 4 with table 5 from their paper, which averages the PER on 5 different languages, trained on 5 hours of data. We observe that overall, we obtained better results, with our best model achieving **28.31%** PER on average on the test set whereas the authors obtain a PER of **37.2%** with their best configuration. However, we did not train our models with only 5 hours of data, some languages are different from those we have used and we do not know the architecture of their network which is maybe more shallow than the one we used. This comparison only allows us to verify the consistency of our results.

4. Conclusions

Through this project, we evaluated three state-of-the-art self-supervised methods on phoneme recognition for various languages, with different proximity to English. We observed that overall, larger models lead to better performances in terms of PER. Also, with our experiments, we concluded using *Base* models that WavLM outperforms Wav2vec2 and for the *Large* models WavLM is better than HuBERT for 3 out of 5 languages and for the 2 others languages the results are close. So, the pretrained method WavLM seems to be better to extract features relevant for phoneme recognition. This is in accordance to the results presented by the authors [7] saying that WavLM is better than HuBERT and Wav2vec2.
With the exception of the Turkish language when fine tuning the models, there is a general trend confirming that

larger amount of training data, as well as more proximity to English improves performances. Moreover, when using frozen features, the discrepancy linked to the Turkish language disappears.
Finally, the amount of training data seems to be logarithmically correlated to the performance of the models. Therefore, when using pretrained models, the advantage provided by increasing the amount of data seems to fade pretty quickly, which means we may not need a large amount of data to obtain decent results.

5. Appendix

5.1. Important links

- https://github.com/ASR-project/Multilingual-PR/tree/main
- https://wandb.ai/asr-project/test-asr?workspace=user-clementapa

5.2. Pretrained Methods' Descriptions

5.2.1 Wav2vec2

This method is essentially based on Transformers [17] architecture and inspired from language models pre-training methods. The architecture of Wav2vec2 is composed of 3 main building blocks:

- Temporal convolutional layers (Causal Convolutions with dilation) that process the raw waveform to get a latent space representation $\mathbb{Z}$.
- A quantization module, that produces quantized representations $\mathbb{Q}$ from $\mathbb{Z}$.
- Transformer layers which creates a contextualized representation $\mathbb{C}$ from the representations Z.

The model learns an optimization objective through contrastive learning, more specifically, given a masked representation z_k, we want to get a context representation c_k such that the latest is as close as possible to z_k, among other possible quantized representations q_j chosen randomly.

5.2.2 HuBERT

HuBERT follows roughly the same structure as Wav2vec2, keeping essential components such as: the convolutional encoders and the Transformers layers. However, the pre-training objective is completely different. At the first training step (*called the discovery phase*), HuBERT generates pseudo-labels by extracting the MFCC feature vectors of the input sequence. K-means algorithm is then applied to these vectors and a label is assigned for each feature vector. Then, these labels are used to generate an embedding, using an embedding layer. For every other training iteration, the CNN encoder is used. The objective of the pre-training step is to predict given a masked input frame, the contextual embedding corresponding of the frame, which should be *close* to the embedding generated by the labels from K-Means. This procedure allows the model to learn MFCC-enhanced context aware embeddings.

5.2.3 WavLM

Compared to Wav2vec2 and HuBERT, WavLM includes masked speech denoising, where some inputs are noisy

and/or overlap. This allows the model to focus also on non-ASR related tasks such as speaker separation, identification and diarization. The authors came up with a contribution on the architecture, by adding the so called *Gated Relative Position Bias* to the Transformers encoder. This method seems to outperform state-of-the-art models on some tasks related to speech processing, including phoneme recognition.

5.3. Statement of contribution

We followed a straightforward planning, which was stated in the project proposal. We started by designing the entire code framework using live coding tools and each focusing on a part of the script when needed. Once goal was to design a framework that would allow us to easily switch between languages and models while offering the possibility to test the best models and reproduce our results. Once this was fully implemented, we decided on the following split :

- Swedish: Arthur
- Russian: Léo
- Turkish: Younes
- Dutch: Clément
- Italian: Everyone

5.4. Results

References

[1] R. Ardila, M. Branson, K. Davis, M. Henretty, M. Kohler, J. Meyer, R. Morais, L. Saunders, F. M. Tyers, and G. Weber. Common voice: A massively-multilingual speech corpus. In *Proceedings of the 12th Conference on Language Resources and Evaluation (LREC 2020)*, pages 4211–4215, 2020. 1

[2] Alexei Baevski, Henry Zhou, Abdelrahman Mohamed, and Michael Auli. wav2vec 2.0: A framework for self-supervised learning of speech representations. *CoRR*, abs/2006.11477, 2020. 1

[3] Mathieu Bernard and Hadrien Titeux. Phonemizer: Text to phones transcription for multiple languages in python. *Journal of Open Source Software*, 6(68):3958, 2021. 1

[4] Lukas Biewald. Experiment tracking with weights and biases, 2020. Software available from wandb.com. 3

[5] Mathilde Caron, Hugo Touvron, Ishan Misra, Hervé Jégou, Julien Mairal, Piotr Bojanowski, and Armand Joulin. Emerging properties in self-supervised vision transformers. *CoRR*, abs/2104.14294, 2021. 1

[6] Guoguo Chen, Shuzhou Chai, Guanbo Wang, Jiayu Du, Wei-Qiang Zhang, Chao Weng, Dan Su, Daniel Povey, Jan Trmal, Junbo Zhang, Mingjie Jin, Sanjeev Khudanpur, Shinji Watanabe, Shuaijiang Zhao, Wei Zou, Xiangang Li, Xuchen Yao, Yongqing Wang, Yujun Wang, Zhao You, and Zhiyong Yan. Gigaspeech: An evolving, multi-domain asr corpus with 10,000 hours of transcribed audio, 2021. 2

[7] Sanyuan Chen, Chengyi Wang, Zhengyang Chen, Yu Wu, Shujie Liu, Zhuo Chen, Jinyu Li, Naoyuki Kanda, Takuya Yoshioka, Xiong Xiao, Jian Wu, Long Zhou, Shuo Ren, Yanmin Qian, Yao Qian, Jian Wu, Michael Zeng, Xiangzhan Yu, and Furu Wei. Wavlm: Large-scale self-supervised pre-training for full stack speech processing, 2022. 1, 4

[8] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics. 1

[9] Hugging Face. Hugging face is an open-source & platform provider of machine learning technologies. 2

[10] Alex Graves, Santiago Fernández, Faustino Gomez, and Jürgen Schmidhuber. Connectionist temporal classification: Labelling unsegmented sequence data with recurrent neural 'networks, 01 2006. 1

[11] Wei-Ning Hsu, Benjamin Bolte, Yao-Hung Hubert Tsai, Kushal Lakhotia, Ruslan Salakhutdinov, and Abdelrahman Mohamed. Hubert: Self-supervised speech representation learning by masked prediction of hidden units, 2021. 1

[12] J. Kahn, M. Riviere, W. Zheng, E. Kharitonov, Q. Xu, P.E. 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. 2

[13] Steven Moran and Daniel McCloy, editors. *PHOIBLE 2.0*. Max Planck Institute for the Science of Human History, Jena, 2019. 1

[14] Vassil Panayotov, Guoguo Chen, Daniel Povey, and Sanjeev Khudanpur. Librispeech: an asr corpus based on public domain audio books. In *Acoustics, Speech and Signal Processing (ICASSP), 2015 IEEE International Conference on*, pages 5206–5210. IEEE, 2015. 2

[15] Morgane Rivière, Armand Joulin, Pierre-Emmanuel Mazaré, and Emmanuel Dupoux. Unsupervised pretraining transfers well across languages, 2020. 2, 4

[16] Steffen Schneider, Alexei Baevski, Ronan Collobert, and Michael Auli. wav2vec: Unsupervised pre-training for speech recognition, 2019. 1

[17] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. *CoRR*, abs/1706.03762, 2017. 5

[18] Changhan Wang, Morgane Rivière, Ann Lee, Anne Wu, Chaitanya Talnikar, Daniel Haziza, Mary Williamson, Juan Pino, and Emmanuel Dupoux. Voxpopuli: A large-scale multilingual speech corpus for representation learning, semi-supervised learning and interpretation, 2021. 2

[19] Jure Zbontar, Li Jing, Ishan Misra, Yann LeCun, and Stéphane Deny. Barlow twins: Self-supervised learning via redundancy reduction. *CoRR*, abs/2103.03230, 2021. 1

Language	Training data (in hours)	Model	PER validation	PER test	Runs
Italian	62.34	Wav2Vec2 *Base*	19.05	17.95	Wav2Vec2_it
		HuBERT *Large*	**14.05**	**12.67**	Hubert_it
		WavLM *Base*	19.83	25.60	WavLM_it
Russian	15.55	Wav2Vec2 *Base*	32.16	31.66	Wav2Vec2_ru
		HuBERT *Large*	25.10	24.09	Hubert_ru
		WavLM *Base*	**20.25**	**18.88**	WavLM_ru
Dutch	12.78	Wav2Vec2 *Base*	16.18	20.83	Wav2Vec2_nl
		HuBERT *Large*	**12.77**	**16.49**	Hubert_nl
		WavLM *Base*	15.96	19.91	WavLM_nl
Swedish	3.22	Wav2Vec2 *Base*	26.50	24.16	Wav2Vec2_sv
		HuBERT *Large*	**21.77**	**19.38**	Hubert_sv
		WavLM *Base*	26.86	24.61	WavLM_sv
Turkish	2.52	Wav2Vec2 *Base*	19.62	19.03	Wav2Vec2_tr
		HuBERT *Large*	**15.51**	**14.19**	Hubert_tr
		WavLM *Base*	19.85	18.95	WavLM_tr
Average	-	Wav2Vec2 *Base*	22.70	22.73	
		HuBERT *Large*	**17.84**	**17.36**	-
		WavLM *Base*	20.55	21.59	

Table 3. Table of experiments when models are **fine tuned**. Here, we compare 3 different pretrained models. The models were fine tuned on the phoneme recognition task with different languages and a varying amount of training data.

Language	Training data (in hours)	Model	PER validation	PER test	Runs
Italian	62.34	Wav2Vec2 *Base*	38.94	36.84	Wav2Vec2_it_tf_freezed
		WavLM *Base*	**27.29**	**25.98**	WavLM_it_tf_freezed
		HuBERT *Large*	23.85	21.15	Hubert_it_tf_freezed
		WavLM *Large*	**21.02**	**18.80**	WavLM_it_tf_freezed
Russian	15.55	Wav2Vec2 *Base*	50.11	48.69	Wav2Vec2_ru_tf_freezed
		WavLM *Base*	**40.66**	**38.76**	WavLM_ru_tf_freezed
		HuBERT *Large*	38.36	36.18	Hubert_ru_tf_freezed
		WavLM *Large*	**34.48**	**32.26**	WavLM_ru_tf_freezed
Dutch	12.78	Wav2Vec2 *Base*	40.15	39.23	Wav2Vec2_nl_tf_freezed
		WavLM *Base*	**34.94**	**35.67**	WavLM_nl_tf_freezed
		HuBERT *Large*	**27.62**	**26.68**	Hubert_nl_tf_freezed
		WavLM *Large*	27.71	27.19	WavLM_nl_tf_freezed
Swedish	3.22	Wav2Vec2 *Base*	50.30	45.23	Wav2Vec2_sv_tf_freezed
		WavLM *Base*	**43.65**	**40.55**	WavLM_sv_tf_freezed
		HuBERT *Large*	37.34	**32.68**	Hubert_sv_tf_freezed
		WavLM *Large*	**37.25**	33.14	WavLM_sv_tf_freezed
Turkish	2.52	Wav2Vec2 *Base*	53.92	52.08	Wav2Vec2_tr_tf_freezed
		WavLM *Base*	**47.18**	**45.53**	WavLM_tr_tf_freezed
		HuBERT *Large*	39.55	37.08	Hubert_tr_tf_freezed
		WavLM *Large*	**30.66**	**30.14**	WavLM_tr_tf_freezed
Average	-	Wav2Vec2 *Base*	46.68	44.41	
		WavLM *Base*	**38.74**	**37.30**	-
		HuBERT *Large*	33.34	30.75	
		WavLM *Large*	**30.22**	**28.31**	

Table 4. Table of experiments using **frozen features**. Here, we compare 4 different pretrained models. The objective was to train a linear layer, using pretrained models' frozen features, on the phoneme recognition task with different languages and a varying amount of training data.

Training set	Training data	Model	PER validation	PER test	Runs
5%	∼ 10 min	Wav2Vec2 *Base*	55.35	50.91	Wav2Vec2_sv_0.05_train_tf_freezed
		HuBERT *Large*	**44.96**	**39.38**	Hubert_sv_0.05_train_tf_freezed
		WavLM *Base*	56.22	51.25	WavLM_sv_0.05_train_tf_freezed
10%	∼ 20 min	Wav2Vec2 *Base*	52.97	49.01	Wav2Vec2_sv_0.1_train_tf_freezed
		HuBERT *Large*	**42.61**	**37.50**	Hubert_sv_0.1_train_tf_freezed
		WavLM *Base*	46.54	43.64	WavLM_sv_0.1_train_tf_freezed
50%	∼ 2 h	Wav2Vec2 *Base*	51.23	46.24	Wav2Vec2_sv_0.5_train_tf_freezed
		HuBERT *Large*	**39.91**	**35.27**	Hubert_sv_0.5_train_tf_freezed
		WavLM *Base*	44.57	42.33	WavLM_sv_0.5_train_tf_freezed
100%	∼ 3 h	Wav2Vec2 *Base*	50.30	45.23	Wav2Vec2_sv_tf_freezed
		HuBERT *Large*	**37.34**	**32.68**	Hubert_sv_tf_freezed
		WavLM *Base*	43.65	40.55	WavLM_sv_tf_freezed

Table 5. Table showing the impact of the amount of training data on 3 different pretrained models when using **frozen features**. The results were obtained on the same language: Swedish.

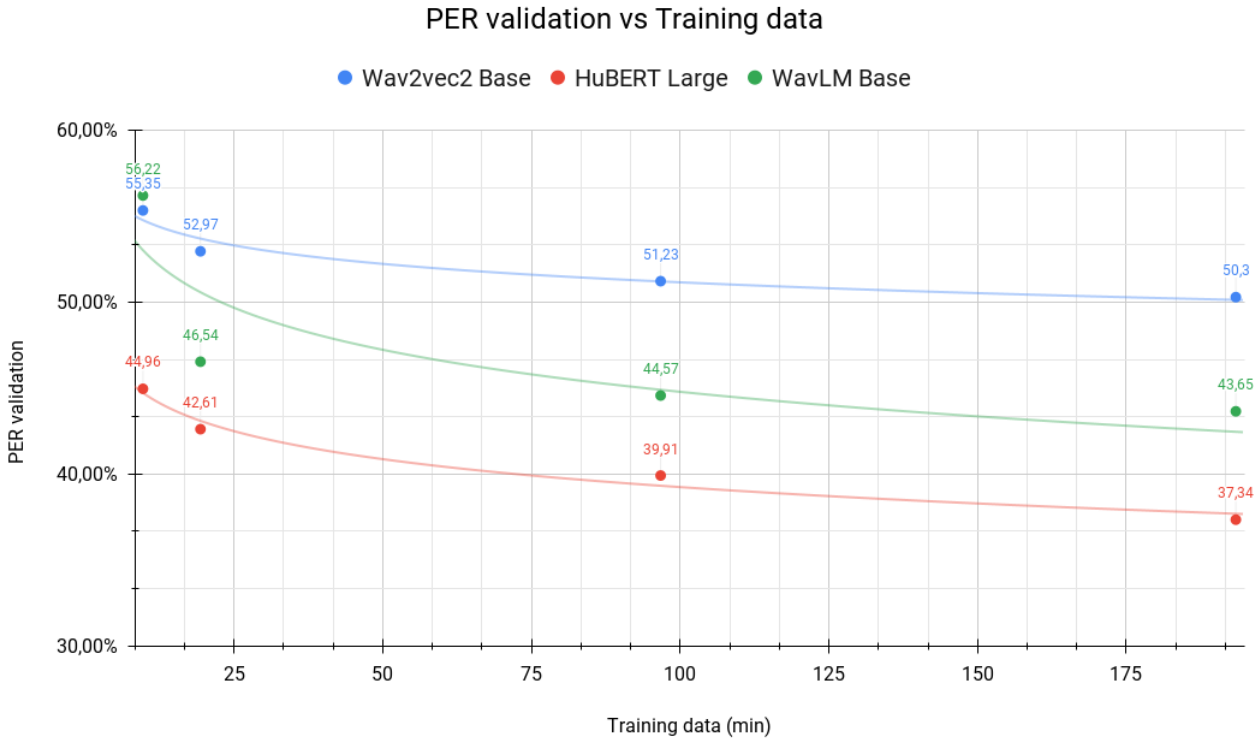

Figure 3. PER on the validation set vs Training data for the Swedish language using frozen features.

Figure 4. PER on the test set vs Training data for the Swedish language using frozen features.