

Multi-lingual Dysphonia Detection and Severity Estimation Using Deep Learning Architectures

Nguyen Vinh Hung – K2KY36
Shende Ayush Nitin - WLIW35

Introduction











This document summarises the “*Multi-lingual Dysphonia Detection and Severity Estimation Using Deep Learning Architectures*” which is Hung – Ayush’s team project in *Deep Learning* class.

This project aims to develop a deep learning model to diagnose dysphonic speech in multilingual scenarios and estimate the severity of the condition.

The GitHub repo can be found here: [GitHub - AyushShende0311/Deep_Learning_Project-Multi-lingual-Dysphonia-Detection](https://github.com/AyushShende0311/Deep_Learning_Project-Multi-lingual-Dysphonia-Detection)

Data source

We are using audio data in multiple languages (Hungarian, English, German, Dutch), categorized by gender and speaker status (healthy or patient). The data is sourced directly to Google Colab from Google Drive. This dataset is substantial and has been sourced from various domains, primarily from the medical testing domain. We have obtained official consent to use the data for this project and we received the data from BME university's database via the project instructor.

 Dysphonia Dutch	 me	Oct 11, 2024	—
 Dysphonia English	 me	Oct 11, 2024	—
 Dysphonia German	 me	Oct 11, 2024	—
 Dysphonia Hungarian	 me	Oct 13, 2024	—
 Dysphonia Portuguese	 me	Nov 16, 2024	—

The following is the characteristic of the data:

- The dataset includes a variety of languages, with an imbalance in the number and length of files across languages.
- German files are shorter and more abundant, while Hungarian, Dutch, English, Portuguese have fewer but longer sequences.
- To address this imbalance, we chunked the longer audio files (Hungarian, Dutch, English, Portuguese) to match the shorter length of German files.

Data processing

As mentioned in the **Data source** part above, we need to chunk the audio files recorded in Hungarian, Dutch, English and Portuguese languages due to the length mismatch.

Chunking the Audio

Audio files in Hungarian, English, Dutch, and Portuguese were chunked to match the length of the German files using the [Chunking_Script.ipynb](#) notebook, which contains the `chunk_audio_in_folder` function.

Speaker Embeddings Extraction

We used **x-vector** model (via *SpeechBrain*), which is a time delay neural network (TDNN) model to extract speaker embeddings for each audio file. The details can be found here:

<https://huggingface.co/speechbrain/spkrec-xvect-voxceleb>.

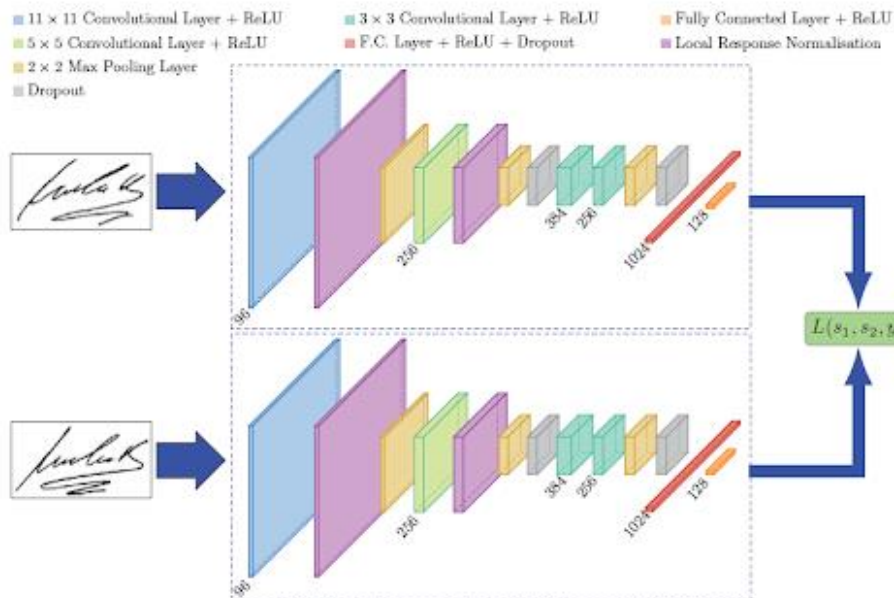
This process is detailed in the *EmbeddingModel.ipynb* notebook, and outputs a CSV file (*embeddings_final.csv*) with the following information:

- File_name: Name of the chunked audio file
- Embedding: Speaker embedding (192-dimensional vector)
- Category: Indicator of healthy or dysphonic voice
- Gender: Speaker's gender
- Language: Language of the audio file
- Speaker_id: ID of the speaker

This CSV file forms the main dataset for training our machine learning model, which will detect specific patterns in the speech and identify the presence of dysphonia.

Method of training

For this project, we use *Siamese Neural Network* (SNN) as our model.



For running the model, to use SNN, pairs of embedding vectors are made which can be either similar or dissimilar (see *TrainingSN.ipynb* notebook - *DysphoniaPairsDataset*), after which we define the SNN architecture and train the model. In specific, two embeddings are considered *similar* if they came from the same speaker, same label (patient or healthy), same gender and same language; otherwise, they are considered *dissimilar*.

The structure of the SNN is as follows: the 1-dimensional convolution is followed by a LeakyReLU (which allows a small gradient for negative values in neural networks), which is followed by a 1-dimensional BatchNorm – we do this 3 times.

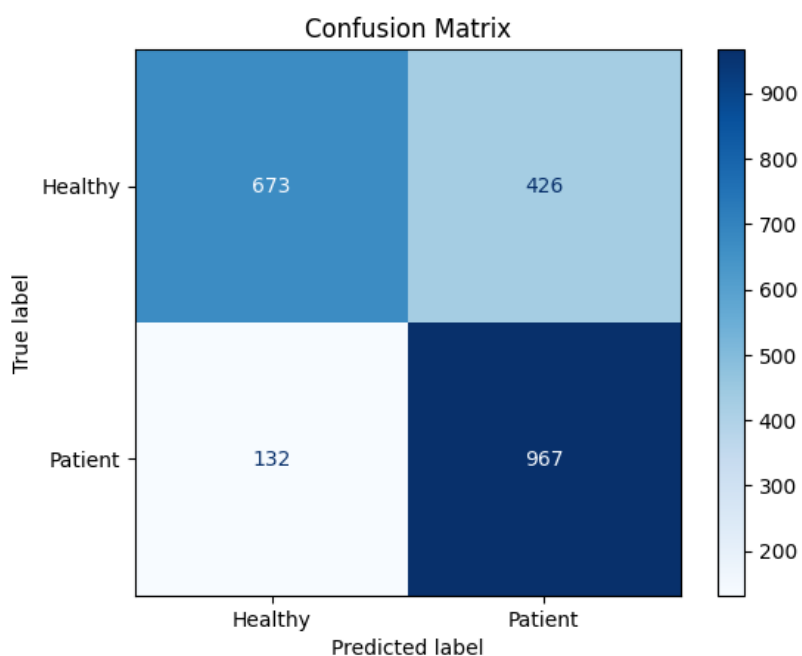
We run the model with 300 epochs, with the initial learning rate of 0.1. However, the learning rate is modified accordingly with *ReduceLROnPlateau*. We aim to minimize the loss metric which is measured by cosine contrastive loss. Contrastive loss takes the output of the network for a positive example and calculates its distance to an example of the same class and contrasts that with the distance to negative examples. Said another way, the loss is low if positive samples are encoded to similar (closer) representations and negative examples are encoded to different (farther)

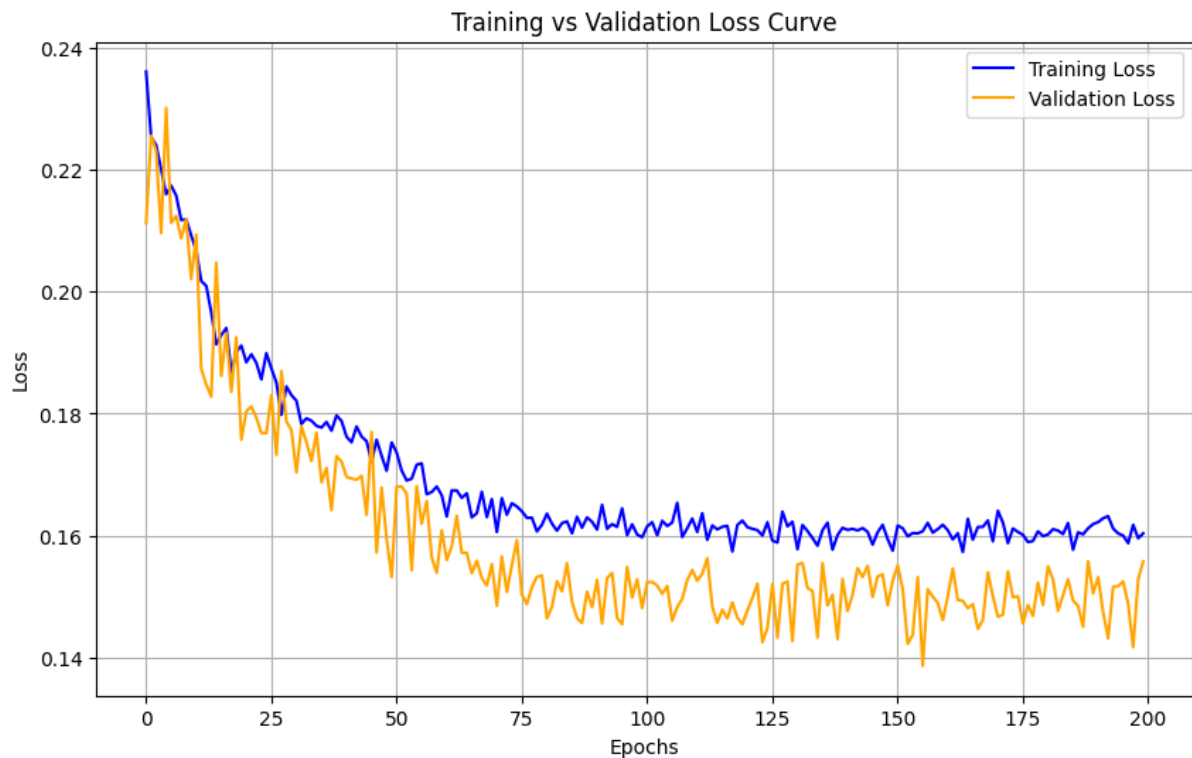
representations. The cosine contrastive loss is the cosine of the contrastive loss. We also applied the dynamic similarity threshold based on the training status (see *dynamic_cosine_threshold* function).

Results

The result of the run can be found as follows, also including the confusion matrix and the loss metric with epochs.

Test Loss: 0.1514
Accuracy: 0.7275
Precision: 0.6766
Recall: 0.8717
F1 Score: 0.7618





Conclusion

We can see that the SNN is a good model for the problem of detecting dysphonic speech in multilingual scenarios and estimate the severity of the condition. Further development is needed to improve the model.

Reference

- <https://builtin.com/machine-learning/siamese-network>
- <https://github.com/sudharsan13296/Hands-On-Meta-Learning-With-Python/blob/master/02.%20Face%20and%20Audio%20Recognition%20using%20Siamese%20Networks/2.5%20Audio%20Recognition%20using%20Siamese%20Network.ipynb>
- https://en.wikipedia.org/wiki/Siamese_neural_network