MAIS 202 – Deliverable 2

Language Detection Model

Muhammad Basel Ahsan, Mohammad Abrar Fuad, & Henry Lefebvre

Brief Description:

Welcome to our Language Detection Model! This project aims to automatically identify the language of a given text or voice note using signal processing, making it a versatile tool for applications such as content filtering, language-specific analysis, and more.

Dataset chosen:

We have finalized on https://huggingface.co/datasets/mozilla-foundation/common_voice_13_0 as our choice of dataset. We chose this because it consists of unique MP3 recordings (as well as text files) with language classifications, which is what we were looking for.

Additionally, many of the 27141 recorded hours in the dataset also include demographic metadata like age, sex, and accent that may help improve the accuracy. Although there are 100+ languages in the dataset, we will be focusing on well-known languages such as English, Mandarin, French, Italian, Arabic, Hindi/Urdu, Russian, and Spanish.

Methodology:

Data Preprocessing: Since the dataset consists of audio files, the preprocessing step will involve converting these MP3 files into spectrograms or MFCCs, which are standard tools for working with audio data. The demographic metadata, such as age, gender, and accent may also be utilized as additional features to enhance the model's performance.

Machine Learning Model: Our goal is to create a model that can identify the language from a given audio clip. A multiclass classification approach where each language represents a class would work well for our purposes. A CNN seems like a suitable architecture for this task, as they're often used for similar audio classification tasks (e.g. music genre classification).

a. Framework and Tools:

Framework: Convolutional Neural Network (CNN) made using Keras.

- Input Layer: 192x192x1 array representing a grayscale image

- Dense Layer: 512, dropout, then **n** where n is the number of languages.

b. Training/Validation/Test Splits, Regularization, and Hyper-parameters:

Dataset Split:

- 70% Training: This provides a substantial amount of data for the model to learn from.

- 30% Validation: To tune hyperparameters and check for overfitting.

- 0% Test: To evaluate the model's performance on unseen data. Will be added later.

Regularization:

- Image preprocessing to put the spectrograms in standard format.
- Normalizing grayscale color values from 0-255 to 0-1

Hyperparameters:

- Learn rate
- Batch size
- Number of epochs to train for
- Model structure / number of neurons per layer / number and structure of hidden layers

c. Validation Methods:

- Confusion matrix
- Validation accuracy
- Testing accuracy

Overfitting/Underfitting:

- Monitor the training and validation loss. If training loss decreases but validation loss stagnates or increases, it's a sign of overfitting. The model currently gets quite overfit past about the third epoch.

- Regularly save model weights and choose the model with the lowest validation loss for final testing.

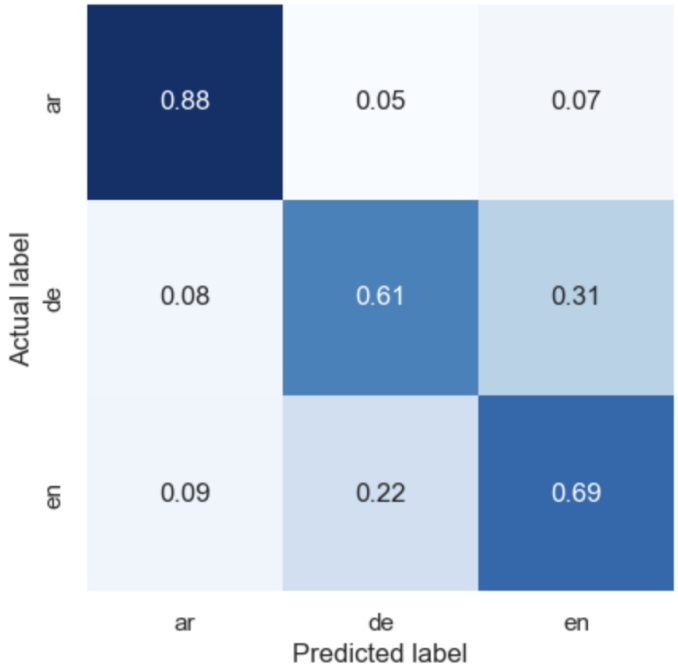
Model Testing: - Evaluate the model on the test set after final training.

- Use metrics such as accuracy, F1 score, and confusion matrix to assess performance across categories. Confusion matrix is useful to tell which languages the model succeeds in distinguishing from one another and whether it's just guessing on others.

d. *Challenges:*

- Overfitting, validation accuracy, maintaining accuracy for a larger pool of languages, data processing and downloading

Preliminary Results:



Next Steps:

Our next steps are to officially begin building and training our network, which will require additional research and learning on our ends before we begin alongside implementing this language identification model into a simple web app with a GUI to record and upload audio or to paste text and then display the predicted language.