AI Project Progress Report

By:
Laura Wilson (lgw0020@auburn.edu)
Coordinator: Daniel Harrison(dah0052@auburn.edu)
Taylor Cross(tac0062@auburn.edu)

Dataset: https://huggingface.co/datasets/renumics/speech_commands_enriched
Title: Speech Command Classification Using Machine Learning Techniques

Problem Description

For this project, we were tasked with training a predictive AI model on a dataset of our choice. We chose to take an audio dataset and train the AI model to classify audio keywords. The complexity of audio signals, influenced by factors such as background noise and speaker variations, are challenges that are difficult to overcome. Addressing them required the development of a machine learning algorithm capable of accurately discerning keywords from audio signals. The aim of this project is to design and implement such an algorithm, utilizing artificial neural networks and feature extraction methods like mel-frequency cepstral coefficients (MFCCs) and chroma features. Through experimentation and evaluation, the effectiveness of the algorithm will be assessed, with the goal of increasing the predictive model's accuracy and precision when classifying audio keywords.

ML algorithm

In this section, we provide a complete description and explanation of the machine learning algorithm implemented to take on the task of audio keyword recognition. The initial step in implementing our machine learning algorithm is extracting the features from the raw audio data. These features are computed using libraries such as Librosa, which provides functions for audio analysis and feature extraction.

We chose various audio features to extract from the raw data to try to get the highest prediction accuracy from our machine-learning model. Mel-frequency cepstral coefficients (MFCCs) are a prominent feature in audio signal processing, calculated through several sequential steps. Initially, the short-time Fourier transform (STFT) of the audio signal is performed to obtain a time-frequency representation. Subsequently, the power spectrum is mapped onto the mel scale utilizing a mel filterbank, which logarithmically compresses the frequency axis to better align with human perception. Following this, the discrete cosine transform (DCT) is applied to the logarithmically-scaled mel power spectrum, resulting in the extraction of MFCCs. Zero Crossing Rate (ZCR) is another crucial feature computed by observing instances where the audio signal changes sign within short time intervals, providing insights into its frequencies and other characteristics. Chroma features, on the other hand, illustrate the distribution of energy across different pitch classes within the audio signal. They are computed by first applying the STFT to the audio signal and then processing the resulting

spectrogram to extract pitch class information. Additionally, the mel spectrogram, which represents the power spectral density of the audio signal on the mel scale, further aids in capturing the frequency distribution over time, contributing to the comprehensive feature set utilized in audio keyword prediction.

Following feature extraction, the subsequent step entails designing the architecture of the neural network model. We opted for a multi-layer perceptron (MLP) architecture, comprising an input layer, multiple hidden layers, and an output layer. The input layer provides the feature vector of size ‘n’ containing the extracted features from the audio data to be used to train the model. Each hidden layer employs a rectified linear unit (ReLU) activation function to introduce non-linearity into the model, increasing the functionality and complexity of the model. The final output layer employs a softmax activation function to convert the input vector into a probability distribution. This allows the model to make predictions based off of the computed probability of each class.

The MLP model is trained using labeled audio data. During training, the model learns to map the input features to the corresponding output labels by adjusting its weights through the process of gradient descent. Training parameters such as the learning rate, batch size, and number of epochs are fine-tuned to optimize performance and prevent overfitting. The training process involves iteratively updating the model parameters based on the gradients of the loss function, which quantifies the amount of error between the predicted and actual labels. We employ the sparse categorical cross-entropy loss function, defined as:

$$L(y,\hat{y}) = -1/N \sum_{i=1}^{N} \sum_{j=1}^{C} y_{ij} \log(\hat{y}_{ij})$$

where ‘N’ denotes the number of samples, ‘C’ signifies the number of classes, y_{ij} represents a binary indicator denoting whether class label ‘j’ is the correct classification for sample ‘i’, and $\hat{y}_{ij}$ signifies the predicted probability that sample ‘i’ belongs to class ‘j’.

The Adam optimizer is an iterative optimization algorithm used to minimize the loss function during the training of our neutral network. This is done by adapting the model parameters as it calculates the learning rates for each parameter based on estimates of first and second moments of the gradients. Early stopping is implemented to prevent overfitting by monitoring the validation loss and restoring the best weights.

Following training, the performance of the MLP model is evaluated on a separate validation dataset to assess its generalization capabilities. Metrics such as accuracy, precision, recall, and F1-score are computed to quantify the model's classification performance across different classes. This evaluation phase helps identify potential areas for improvement and refinement in the model architecture or training procedure.

Once we got our model training on the dataset with only a few of our features, our initial accuracy and precision metrics were only showing a 1% rate in correct classification for our model. However, once we got all of the features extracted correctly and the model was utilizing all of the features to make predictions, our metrics started showing 50% classification

correctness. We then started changing various hyper-parameters of our algorithm to try to increase the performance of our model, such aspects include the depth of our model, adjusting learning rates, optimizing the number of epochs, and even adding additional features like Chroma.This process of model development and refinement allowed for continuous improvement in the accuracy of the machine learning algorithm for predicting audio keywords. Through careful experimentation and analysis, the implementation of the MLP algorithm demonstrates efficiency in classifying audio data.

Tutorial of software

For our software, we used Google Colab which allows 12.67 GB of RAM and at least 10 GB of that RAM will be used from our code. How to use our code is simple as each cell that is present builds onto each other so executing one by one will work and show everything that is needed. But I'll explain what to expect from each cell and what the time limit should be.

The first cell installs the dataset of audio from our dataset with a date rate limit of 1.0e12. You may have a pop-up about restarting the session to install a new version of a package. When this happens you will need to restart the session to ensure that what is needed is installed. This will only take less than a minute to execute.

In the next cell, you will import the needed packages as well as loading and defining the dataset of the training, validation, and testing examples. This will only take about a minute to execute.

The next cell is defining a sample of silence which takes no time to execute.

After that this cell initializes arrays of labels and audio data while loading the audio files from bytes and creating a data frame from the lists that will take about 1-2 minutes to execute.

The cell after is where the features we are using, MFCC, Zero Crossing Rate, Chroma Features, and Mel Spectrogram, are extracted. This will take no time as it is assigning features to named variables.

After extracting the features the cell executed next will apply the features extracted to x and y values of the training, validation, and testing examples this will take the most time at 1 - 2 hours.

The next cell will print the shape of each testing and validation training of x and y values. Which takes no time at all to execute.

The next cell builds an MLP model. It is executed in a few seconds as it has to go through the imports and create it.

The next cell calculates the loss, accuracy, validation loss, and validation accuracy in batches of 64. The next cell will show the validation and training loss and accuracy against each other. The execution of this will take around 10 minutes as it has to go from 1 to 50 epochs.

The next cell extracts values from history objects and plots the training and validation in loss and accuracy of the data. This takes no time at all to execute.

The next cell is calculating precision, recall, and f1 scores. This is executed in a few seconds.

The last cell is calculating and showing the confusion matrix.

Evaluation metrics

The metrics you've chosen—accuracy, cross-entropy loss, F1 score, precision, and recall—are all highly relevant for evaluating the performance of a classification model, such as the MLP model you're developing. Each of these metrics offers unique insights that contribute to a comprehensive assessment of your model under different conditions and for various purposes:

Accuracy: It provides a quick snapshot of your model's overall effectiveness by measuring the proportion of correctly predicted instances. It's particularly useful for giving you a high-level overview when classes are balanced, enabling you to gauge how often the model is correct in general.

Cross-Entropy Loss: This metric is essential during the training phase of your model. By quantifying the difference between the predicted probabilities and the actual binary labels, it serves as a feedback mechanism for optimizing the model's weights through backpropagation. Minimizing this loss directly correlates to improving the model's accuracy in probabilistic terms, making it a critical metric for model development.

F1 Score provides a more nuanced view of model performance, especially in the presence of imbalanced classes. It is calculated as the harmonic mean of precision and recall, incorporating both false positives and false negatives into its computation. The F1 score is particularly valuable because it balances the trade-offs between precision and recall, offering a single metric that can summarize model performance in scenarios where avoiding false positives and false negatives is crucial.

Precision measures the accuracy of positive predictions made by the model. Formally, it is the ratio of true positives (correct positive predictions) to the combined total of true positives and false positives (all positive predictions). Precision is especially important in contexts where the cost of making a false positive error is high, such as in medical diagnostics or fraud detection.

Recall, also known as sensitivity, assesses the model's ability to correctly identify all relevant instances within the dataset. It is calculated as the ratio of true positives to the sum of true positives and false negatives (all actual positive cases). High recall is essential in situations where missing a positive instance (a false negative) carries significant consequences, such as in disease screening where failing to identify a condition could be detrimental.

Overall Accuracy: The model achieved an overall accuracy of 73.42%. This indicates that in approximately 73 out of 100 cases, the model correctly classified the audio keyword. This level of accuracy is commendable given the complexity of audio processing, but there's room for improvement, especially in handling more diverse or noisy data.
Precision: The precision of the model stands at 75.11%. This metric is particularly crucial because it reflects the proportion of positive identifications that were actually correct. A precision of over 75% suggests that the model is relatively reliable in its positive classifications, reducing the likelihood of false positives—a significant factor in many practical applications such as voice-activated control systems where erroneous activations can be disruptive.

Recall: With a recall of 72.65%, the model demonstrates its capability to identify a substantial proportion of all relevant instances within the dataset. While this shows the model's effectiveness in detecting positive cases, there is a notable gap where about 27% of actual positives are not captured. Enhancing recall is essential in scenarios where missing a positive detection can have serious implications, such as in security or emergency response systems.
F1 Score: The F1 score, calculated at 73.12%, balances precision and recall, providing a single measure to gauge the model's accuracy regarding false positives and false negatives. An F1 score closer to the precision and recall values indicates a balanced performance between these metrics. However, aiming to increase this score would ensure a more robust model, especially beneficial in handling imbalanced datasets.
By leveraging these metrics collectively, we gain a comprehensive understanding of our model's performance. Each metric highlights different aspects of prediction accuracy and error types, enabling us to fine-tune our model and address specific issues related to precision, recall, or overall accuracy. This thorough evaluation helps in ensuring that our model not only predicts accurately but also aligns with the specific needs and constraints of the application for which it is being developed.

In conclusion, this project aimed to develop and assess a predictive AI model tailored for the classification of audio keywords. The challenges inherent in audio signal processing, such as background noise and variations in speaker attributes, necessitated the adoption of a sophisticated machine learning approach involving feature extraction techniques and neural network architectures. Utilizing Mel-frequency cepstral coefficients (MFCCs), chroma features, and other relevant audio signal attributes, we constructed a multi-layer perceptron (MLP) model capable of processing and classifying complex audio data.

The model was rigorously evaluated using a suite of metrics including accuracy, cross-entropy loss, F1 score, precision, and recall, each providing unique insights into the model's classification performance. Initial results demonstrated modest classification abilities, which substantially improved with iterative refinements of the feature set and model hyperparameters. The use of these diverse metrics facilitated a balanced evaluation, highlighting strengths in general accuracy and specific areas needing improvement in precision and recall.

Potential improvements for future work could focus on several areas to enhance model performance and applicability. Firstly, incorporating a broader range of audio features and exploring advanced feature extraction methods might capture more nuanced information from audio signals, potentially increasing classification accuracy. Secondly, experimenting with different neural network architectures, such as convolutional neural networks (CNNs) or recurrent neural networks (RNNs), could offer improvements in handling the temporal dynamics of audio data. Another algorithm that could be used is called audio transformers.

Through continuous development and refinement, the pursuit of more sophisticated machine learning strategies promises further advancements in the field of audio keyword classification, paving the way for more accurate and reliable predictive models.