Intent Detection with Audio Classification

Sepehr Alemzadeh
Politecnico di Torino
Student ID: s314315
s314315@studenti.polito.it

Afsoun Abbasi
Politecnico di Torino
Student ID: s314427
s314427@studenti.polito.it

***Abstract*—As one of the cornerstones of Natural Language Processing (NLP), intent detection is a process of understanding the user's purpose or goal when interacting with a system. In this project, we introduce a methodology to classify speech commands for intent detection from human's recorded audio. We begin by an exploratory analysis through the given dataset, perform data pre-processing, choose a model based on deep learning techniques and compare the results of the model on the processed dataset. We base our analysis on a Mel Frequency Cepstral Coefficients (MFCC) spectogram technique to utilize the useful characteristics of both time- and frequency- domains as the features of the Machine Learning model. Our experimental results show that such approach accompanied by the VGG16 model for image classification, gives the best accuracy among all options. We will also provide the relevant plots as well a comparison table showing the superiority of VGG16 with respect to other 1d or 2d Convolutional Neural Networks (CNN) models.**

I. Problem Overview

Intent detection is the process of realizing user's purpose when interacting with a system. This technology that has been around for some time, but has recently become more popular as different organizations look for ways to improve customer experience and engagement. Intent detection is used in a variety of applications, such as Natural Language Processing (NLP), customer service and marketing. For instance, intent detection can be used to lift customer's engagement by understanding the user's actual intent. Therefore, organizations can provide more personalized content and better target their marketing efforts. Additionally, intent detection can be used to improve customer service by providing more accurate and timely responses to inquiries [1].

We consider a multi-class classification problem in order to classify given audio input into a variety of actions and objects identified from the given audio signals. Employing Machine Learning (ML) techniques for intent detection from audio comes with its own unique challenges. One of these challenges is the interpretation of audio signal and use them as features for model training. In this report, we will provide in detail our approach to this classification problem and the schema of how we tackle the raw data obtained from audio files towards time and frequency domains. We then continue the effort by choosing a Neural Networks (NN)-based model for classification.

A. Dataset

The given data is a collection of audio recordings of utterances of actions and objects from various people. The goal of this project is to predict both the action requested and the object that is affected by the action. The dataset consists of a collection of audio files in WAV format divided into two CSV files:

- Development set: a CSV file that contains 9854 recordings with given labels. This set is mainly used for model training and validation as will be discussed in following chapters.
- Evaluation set: a CSV file that contains 1455 recordings without labels. This will be used for evaluating the trained model on an unseen set of data. The results of the evaluation can be obtained through the online leaderboard platform.

Each row of data on these sheets provide several attributes including the path of the file, the speaker ID, the labels (action + object) of the audio, and some additional information on the speaker such as their fluency level as well as their native language and gender. The data is distributed into different directories for each speaker based on their ID and the number of recordings vary among them. The sample width is of 16 bits for all files. Most of the recordings are sampled at 16 KHz, however, there are a few cases with different sampling ratio which result in significantly distinct duration of the audios. This is shown in the form of a histogram in figure 1. To overcome this issue, we process each recording such that the length of each audio signal is equal for all data point. We will discuss this more in the upcoming section.

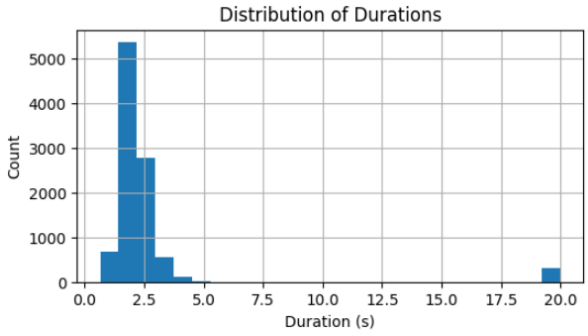

Fig. 1. Distribution of duration of the recordings.

Next, to get some understanding of the provided data, we randomly sample one data point (in this case the very first data available) and look into the time and frequency domain

plots. The time and frequency domain signals are respectively provided in figure 2 and figure 3.

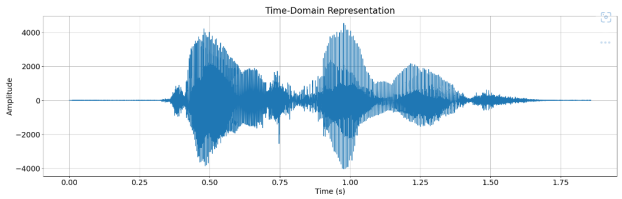

Fig. 2. Audio Representation in Time-Domain.

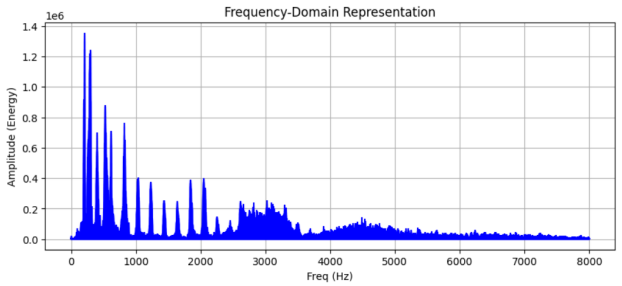

Fig. 3. Audio Representation in Frequency-Domain.

As mentioned earlier, figure 2 verifies that there are some blank silent audio on the tails of the recording making recording inconsistent in duration. Also note that the amplitude remains in the range $[-32,768, 32,768]$ which follows as a result of 16 bits sample width.

II. Proposed Approach

A. Preprocessing

First, to make the dataset consistent we set a threshold of 5 seconds and for those recordings with duration more that this threshold, we cut the audio down to 5 sec, and for those that are shorter, we add silence (in form of appended 0 values to the time-domain signal) such that the duration of the recording becomes 5 sec. Then, we decide to take advantage of both time and frequency domain as they are both valuable sources for feature engineering phase of the ML model design. We conduct three experiments with different models where we use max-pooling convolutional layers technique to reduce the size of feature maps. In CNN's max-pooling layers, feature maps are divided into rectangular (sub)regions, and the features in each rectangle are down-sampled to a single value, by taking their maximum value. Figure 4 depicts an example of 2D max-pooling in CNNs for a simple given 2D-array [2].

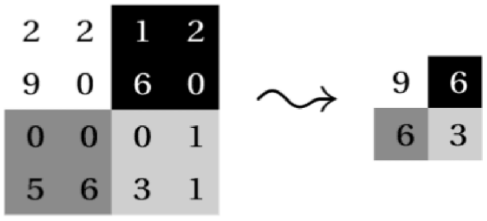

Fig. 4. An example of 2D max-pooling.

In the first experiment, we use 1D convolutional layers on only the time-domain signal and leave out the frequency-domain information in our feature set. In the next two set of experiments, we extend the same method to 2D max-pooling on the MFCC spectogram plots obtained from both time and frequency domains. Mel-frequency cepstral coefficients (MFCCs) are coefficients that make up a representation of the short-term power spectrum of a sound based on a linear cosine transform of a log power spectrum on a nonlinear mel-scale of frequency [3]. MFCC is derived by the Fourier transform of an audio signal mapped on to a mel-scale. Figure **??** show the MFCC signal of the same audio file we used in figures 2 and 3 after being augmented with zeros to get to the 5 sec threshold.

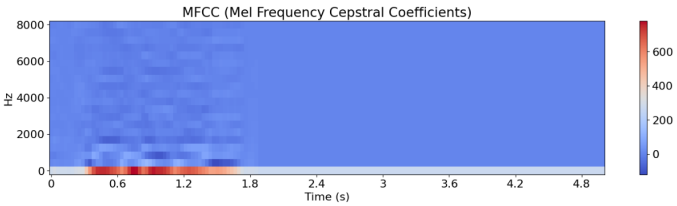

Fig. 5. MFCC Representation of a Sample Audio Signal.

The MFCC is basically a 2D matrix and max-pooling as discussed above, helps collecting features out of this array.

The other thing to note is that the labels in the dataset are in text format (concatenated actions and objects) so we need to convert them into class numbers. To this end, we create a list of the class labels using on the dataset and then use the class label lists index number to assign a number to the text labels.

B. Model Selection

For this project, we use CNNs as a deep learning technique that are a specialized type of artificial neural networks that use a mathematical operation called convolution in place of general matrix multiplication. CNNs are specifically designed to process pixel data and are proved to be a powerful tool in image analysis applications such as image recognition and processing. As any other NN architecture, a CNN consists of an input layer, hidden layers and an output layer. In any feed-forward NN, the middle layers are called hidden as their inputs and outputs are masked by the activation function and final convolution. In a CNN, the hidden layers include layers that perform convolutions. Typically this includes a layer that performs a dot product of the convolution kernel with the layer's input matrix. The activation function is commonly ReLU and as the convolution kernel slides along the input matrix for the layer, the convolution operation generates a feature map, which in turn contributes to the input of the next layer. As introduced earlier, this is followed by other layers such as pooling layers, fully connected layers and normalization layers.

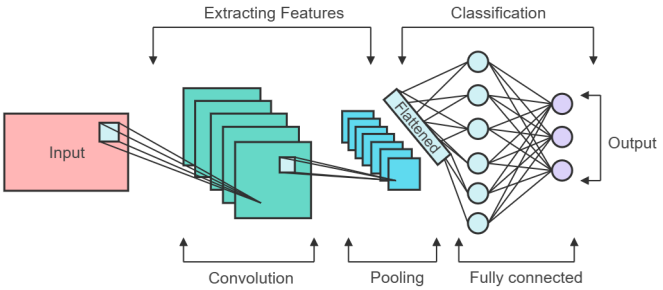

Fig. 6. Schematic of a sample CNN architecture.

In a CNN, the input is a tensor with a shape $n_i \times h_i \times w_i \times c_i$ where n_i, h_i, w_i and c_i respectively denote number of inputs, input height, input width, input channels. After passing through a convolutional layer, the image turns into a feature map with shape $n_i \times h_f \times w_f \times c_f$ where h_f, w_f and c_f correspond to feature map height, feature map width, feature map channels. Convolutional layers affect the input and pass its result to the next layer.

Note that although fully connected feed-forward neural networks can be used to learn features and classify data, this architecture is generally impractical for larger inputs such as high-resolution images as it would require a very high number of neurons, even in a shallow architecture, due to the large input size of images, where each pixel is a relevant input feature. Instead, convolution reduces the number of free parameters, allowing the network to be deeper.

Also, for regularization, we use the *dropout* method to reduce over-fitting. At each training stage, individual nodes are either dropped out (ignored) with some probability or kept, so that a reduced network is left. The removal also contains incoming and outgoing edges to a dropped-out node. Then, only the reduced network is trained on the data in that stage and the removed nodes are placed back into the network with their original weights. By reducing node interactions, dropout leads the layers to learn more robust features that better generalize to new data. The method also significantly improves training speed.

C. Hyper-parameter Selection

The hyperparameters of our algorithm are mainly due to the NN model requirements. We set the batch size to $n_b = 32$ and split the training data such 90% are allocated to training and 10% to validation. With the dataset in hand, this projects to 8,869 number of training audio files and 985 validation audio files. The optimization method is set to Adam with learning rate $\ell_r = 0.001$. The learning rate is scheduled to damp with a linear strategy using PyTorch's learning rate scheduler function. Since we have a classification problem, the *cross-entropy* loss function is picked. For all forward passes of network we choose a max pooling strategy with ReLU activation and dropout for regularization.

III. Results

For experimentation, we choose three different models. The first model would be a basic 1D Convolution that gets as input only the time-domain signal and the next ones possess a more advanced architecture. In what follows, we describe the architecture of each model:

- **Model 1:** Contains 4 1D-convolutional layers all with stride equal to 1 and input channels of $1, 8, 16, 32$ and output channels of $8, 16, 32, 64$ respectively. The probability of dropout is set to 0.3. The convolutional layers are followed by 3 fully connected layers of size 62976, 256 and 128. As expected, the output layer contains 7 bins which corresponds to the number of classes (labels).

- **Model 2:** Includes 2 2D-convolutional layers with stride 1 and input size of $1, 8$ and output size of $8, 16$. The probability of dropout is set to 0.3. The convolutional layers are followed by 4 fully connected layers of size 2064, 1024, 256 and 128 respectively. Note that, while we use 2D-convolutional layers comparing to Model 1 with 1D-convolutional layers, the network is shallower than the one in model 1.

- **Model 3:** For the last model, we use the well-established and popular VGG16 model which is significantly higher in complexity and number of layers in the network. VGG16 was initially proposed in the paper [4] which won the ILSVRC competition in 2014. The first two layers have 64 channels of a 3×3 filter size and the same padding. Then after a max pool layer, two layers have convolution layers of 128 and filter size 3×3. This is followed by a max-pooling layer of stride 2×2. Then there are 2 convolution layers of filter size 3×3 and 256 batch normalization filters. After that, there are 2 sets of 3 convolution layers and a max pool layer. Each has 512 batch norm filters of 3×3 with the same padding. This image is then passed to the stack of two convolution layers. In these convolution and max-pooling layers, the filters we use are of the size 3×3. There is a padding of 1-pixel (same padding) done after each convolution layer to prevent the spatial feature of the image. After this there is 3 fully connected layers, the first layer takes input from the last feature vector and outputs a vector of length 4096, the second layer also outputs a vector of size 4096, but the third layer output a 7 channels for 7 classes of the problem in hand, therefore, the third fully connected layer is used to implement softmax function to classify 7 classes. All the hidden layers use ReLU as its activation function. The schema of VGG16 is provided in the following:

Fig. 7. VGG16 Architecture.

The result of the training for each model is provided as follows:

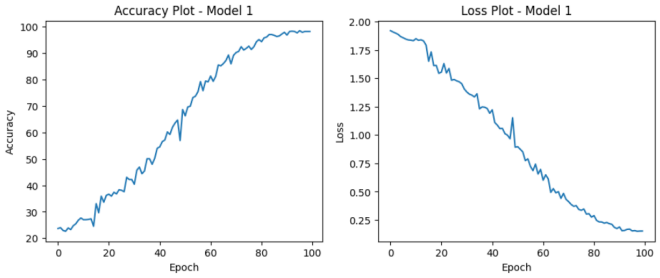

Fig. 8. Results of Model 1 Training.

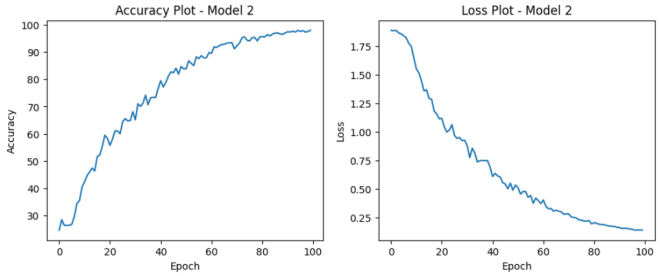

Fig. 9. Results of Model 2 Training.

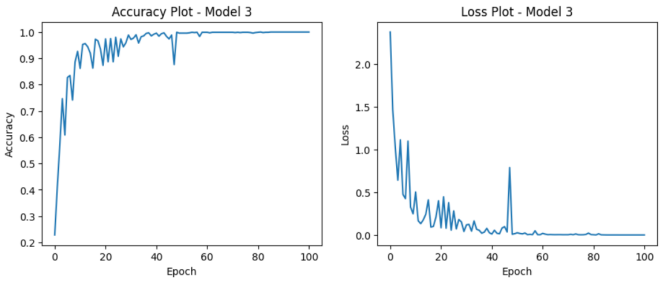

Fig. 10. Results of Model 3 Training.

The accuracy and loss of training besides the results on test data (submitted on class leaderboard) are provided in the following table:

	Train Loss	Train Acc	Test Acc
Model 1	0.1531	98.22%	66.9%
Model 2	0.1374	98.01%	74.3%
Model 3	0.0000	100.00%	99.3%

TABLE I
RESULTS OF ALL THREE SET OF EXPERIMENTS.

IV. DISCUSSION

In this project, we used Convolutional Neural Networks models to classify audio signal based on the intent of the speaker. We analyzed raw wave form of the given audio files and performed some data pre-processing to make the data machine readable and compatible with Machine Learning models. We conducted some exploratory analysis to show how useful it can be to have a combination of time and frequency domains simultaneously, rather than only one of the two. We then decided to use MFCC spectogram as a signal which contains the best of the two ends. We then continued with the Machine Learning model design and trained 3 different Neural Networks architecture. Finally, our results show that while simple model architectures can give good accuracy, moving towards more advanced designs is the key to getting very high accuracy numbers. Also, our efforts show that a deeper Neural Network model does not necessarily lead to a better results, since the design parameters and hyper-parameters are also important.

The results obtained show promising accuracy and not much room for improvement. However, the reader should note that the current classification problem is not considered as highly complex since the input variables and the number of classes are quiet limited. As a side reminder, the VGG16 model has won the ILSVRC competition with 1000 classes which is hugely more advanced than audio recognition for 7 output classes. Therefore, we believe that simpler Neural Network architectures (than VGG16) should give high enough accuracy.

REFERENCES

[1] Camastra F, Vinciarelli A. Machine learning for audio, image and video analysis: theory and applications. Springer; 2015 Jul 21.
[2] Krizhevsky A, Sutskever I, Hinton GE. Imagenet classification with deep convolutional neural networks. Communications of the ACM. 2017 May 24;60(6):84-90.
[3] T. Ganchev, N. Fakotakis, and G. Kokkinakis, Comparative evaluation of various mfcc implementations on the speaker verification task," in Proceedings of the SPECOM, vol. 1, no. 2005, pp. 191–194.
[4] Simonyan, K., Zisserman, A. (2014). Very Deep Convolutional Networks for Large-Scale Image Recognition. CoRR, abs/1409.1556.