# Spoken Digit Recognition (Audio Classification)

**Introduction**

The Spoken Digit is a free audio dataset of spoken digits.

A simple audio/speech dataset consisting of recordings of spoken digits in wav files at 8kHz. The recordings are trimmed so that they have near minimal silence at the beginnings and ends.

6 speakers - 3000 recordings (50 of each digit per speaker) English pronunciations.

The problem statement is of reducing the background noise as much as possible, making the digit sound clearly, recognise the digit and classify it as one of the 10 classes (0 to 9) even if the pitch of the speaker is low.

**Meta Data of Data set**

The dataset Spoken Digits consists of 3000 voice recordings of three speakers namely, “Jackson”, “Nicolas”, “Theo”, “Yweweler”, “George”, “Lucas”.

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Gender | Accent | Language |
| Nicolas | Male | BE/French | English |
| Theo | Male | USA/neutral | English |
| Jackson | Male | USA/neutral | English |
| Yweweler | Male | DEU/German | English |
| George | Male | GRC/Greek | English |
| Lucas | Male | DEU/German | English |

The file name format is as follows:  
 {digitLabel}\_{speakerName}\_{index}.wav Example: 0\_george\_2.wav

**Project Objectives**

We are going to build and train the model to recognize spoken digits.

This can be used in automation of several tasks like setting AC temperature, controlling TV, etc.

The objective of the project is to recognize the digit spoken by the speaker, irrespective of his/her gender or accent, using appropriate deep learning models.

**Procedure**

The data is in the form of audio clips. These audio clips are converted into spectrograms.   
Spectrogram is a visual representation of the spectrum of frequencies of a signal as it varies with time. This is like fingerprint of humans (unique).

**Phase I**

**Exploratory Data Analysis**

Librosa and IPython libraries are imported for EDA. We use the load function of the librosa library to load the data, sample rate of the audio file. The waveplot function of the librosa. Display library is used to plot the wave diagram of the audio file with the sample rate of the audio file.

The ipd.Audio function is used to get the audio of the audio file.

Sample\_rate refers to the number of samples of audio recorded every second. Using Librosa, we get the sampling rate as 22050 which means that every second, 22,050 samples were taken.

**Preprocessing**

The function **“wav\_to\_spectogram”** is defined, which converts the audio forms in the form of wav files to spectrograms taking the path of the audio files and the path of the spectrogram folder where the converted file is to be saved. This is defined for a single file.

Here the size of the figure is set in inches, axes is added and the locator is set for X and Y axis.

The function **“dir\_to\_spectogram”** is defined, which converts the audio files in the recordings folder to spectrograms and stores it in the spectrograms folder.

Then, the spectrograms are divided into train and test sets based on the index after splitting the file name into 3 parts, the digitLabel, speakerName and the index.

The images and labels are separated in the train and test set. The list is to converted to numpy array to define input shape. The y\_train and y\_test are converted to one hot representation where the correct output is labelled 1 and all the others are labelled 0.

Finally, the images are normalized by dividing it by 255.

**Phase II and III**  
We use Keras for the model building.

I created 5 models, 4 of which are CNN Models and the other one is an LSTM Model.

**Model – 1 – CNN Model with 3 CNN Layers**

* **Model Hyperparameters**
  1. Optimizer - Adadelta
  2. Activation - ReLU
  3. Number of epochs - 100
  4. Batch Size - 50
  5. Learning rate - Adadelta default
  6. Loss - Categorical Crossentropy
* **Model Structure**
  1. 3 convolutional layers
  2. 1 Max Pooling Layer
  3. 3 dense layers (MLP)
  4. Softmax Activation for output
  5. BatchNormalization Layer after every Conv Layer and Dense Layer.
  6. Dropout for every layer of MLP.

**Model – 2 – CNN Model with 2 CNN Layers**

* **Model Hyperparameters**
  1. Optimizer – Adam
  2. Activation - ReLU
  3. Number of epochs - 100
  4. Batch Size - 50
  5. Learning rate – 0.001
  6. Loss - Categorical Crossentropy
* **Model Structure**
  1. 2 convolutional layers
  2. 1 Max Pooling Layer
  3. 3 dense layers (MLP)
  4. Softmax Activation for output
  5. BatchNormalization Layer after every Conv Layer and Dense Layer.
  6. Dropout for every layer of MLP. (Dropout with a different value)

**Model – 3 – CNN Model with 2 CNN Layers and EarlyStopping**

* **Model Hyperparameters**
  1. Optimizer – Adam
  2. Activation - ReLU
  3. Number of epochs - 100
  4. Batch Size - 50
  5. Learning rate – 0.001
  6. Loss - Categorical Crossentropy
* **Model Structure**
  1. 2 convolutional layers
  2. 1 Max Pooling Layer
  3. 2 dense layers (MLP)
  4. Softmax Activation for output
  5. No Batch Normalisation
  6. Dropout for every layer of MLP.

**Model – 4 – CNN Model with 6 CNN Layers**

* **Model Hyperparameters**
  1. Optimizer – Adam(RMSprop)
  2. Activation - ReLU
  3. Number of epochs - 100
  4. Batch Size - 50
  5. Learning rate – 0.001
  6. Loss - Categorical Crossentropy
* **Model Structure**
  1. 6 convolutional layers
  2. 2 Max Pooling Layers
  3. 2 dense layers (MLP)
  4. Softmax Activation for output
  5. No Batch Normalization
  6. Dropout for every layer of MLP.

**Model – 5 – LSTM**

* **Model Hyperparameters**
  1. Optimizer – Adam
  2. Number of epochs – 100
  3. Batch Size – 50
  4. Learning rate – 0.001
  5. Loss - Categorical Crossentropy
* **Model Structure**
  1. 1 dense layer
  2. Softmax Activation for output

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | Model 1 | Model 2 | Model 3 | Model 4 | Model 5 |
| Optimizer | Adadelta | Adam | Adam | Adam(RMSprop) | Adam |
| Number of Epochs | 100 | 100 | 100 | 100 | 100 |
| Batch Size | 50 | 50 | 50 | 50 | 50 |
| Learning Rate | Adadelta Default | 0.001 | 0.001 | 0.001 | 0.001 |
| Loss | Categorical Crossentropy | Categorical Crossentropy | Categorical Crossentropy | Categorical Crossentropy | Categorical Crossentropy |

**MODEL HYPERPARAMETERS – SUMMARY**

**MODEL STRUCTURE – SUMMARY**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | Model 1 | Model 2 | Model 3 | Model 4 | Model 5 |
| No. of CNN Layers | 3 | 2 | 2 | 6 | - |
| No. of Max Pooling Layers | 1 | 1 | 1 | 2 | - |
| No. of Dense Layers | 3 | 3 | 2 | 2 | 1 |
| Output Activation | Softmax | Softmax | Softmax | Softmax | Softmax |
| BatchNormalisation | After every Conv and Dense Layer | After every Conv and Dense Layer | - | - | - |
| Dropout | For every layer of MLP | For every layer of MLP | For every layer of MLP | For every layer of MLP | - |

**Phase-IV**

Performance Analysis:

For the accuracy metric, comparison among the 5 models is as follows:

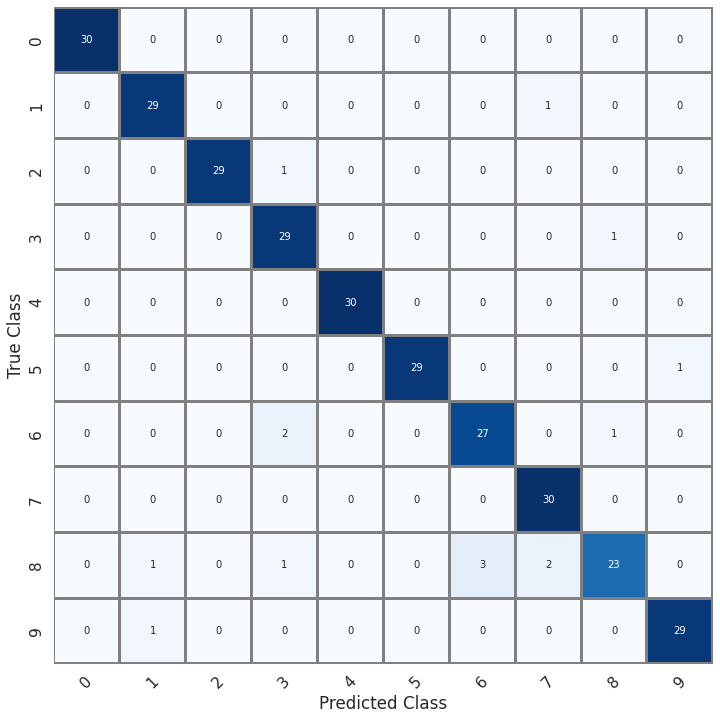
**Model – 1 – CNN Model with 3 CNN Layers : 95%**

**Chart, line chart

Description automatically generatedChart, line chart

Description automatically generated**

**Confusion Matrix | Model 1**

****

We can see some of the classes being wrongly predicted in the small 300 samples.

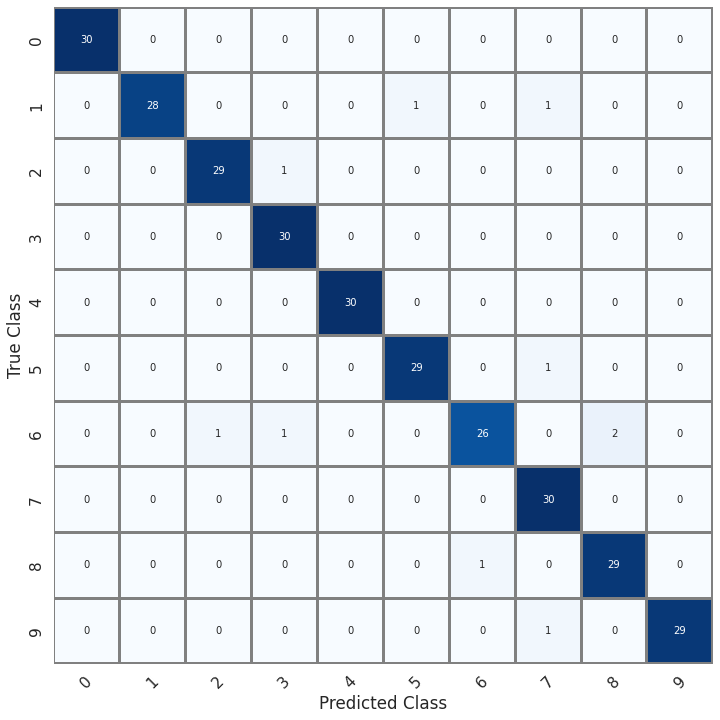
**Model – 2 – CNN Model with 2 CNN Layers : 96.66%**

**Chart

Description automatically generatedChart

Description automatically generated**

**Confusion Matrix | Model 2**

****

We can see some of the classes being wrongly predicted in the small 300 samples data.

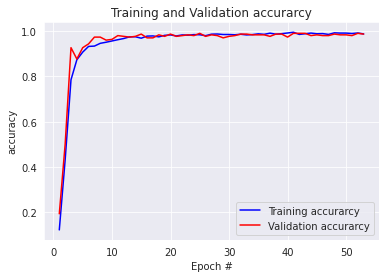
**Model – 3 – CNN Model with 2 CNN Layers (EarlyStopping) : 96.66%**

**Chart

Description automatically generatedChart, line chart

Description automatically generated**

**Model – 4 – CNN Model with 6 CNN Layers: 98.66%**

**Chart

Description automatically generated**

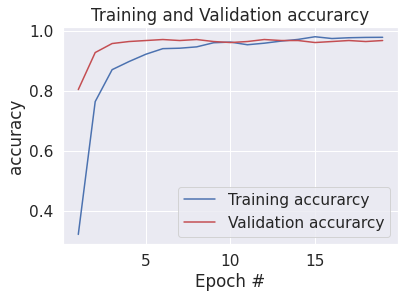
**Confusion Matrix | Model 4**

**Chart, scatter chart

Description automatically generated**

We can see that there are hardly any misclassifications, as the accuracy is very high and there isn’t overfitting as well. This makes it the best model.

**Model – 5 – LSTM: 92.33%**

**Chart

Description automatically generated**

**Conclusion**

We can clearly observe that the accuracy of the CNN Model **with 6 CNN Layers and the hyperparameters** as mentioned before is the highest. It has an **accuracy of 98.66%**. We changed the number of convolutional layers, pooling layers, dense layers, dropout layers and optimizer to achieve this.

Thus, the CNN model with 6 CNN Layers is the best model for spoken digit recognition for this dataset, according to my analysis.