JSpeech

JSpeech is a company that specializes in developing advanced speech recognition technology for recognizing a single speech command. With their cutting-edge software and algorithms, **JSpeech** offers a range of solutions for businesses and individuals who require accurate and reliable voice recognition technology. Whether you need to control your smart home devices, operate a hands-free system in your car, or give voice commands to a robot, **JSpeech's** technology can help you achieve your goals efficiently and effectively.

JSpeech's team of developers has decided to bring on board an enthusiastic programmer, which happens to be you, to increase efficiency in the company. Your task as a programmer is to develop a program using **Python** language and **Tensorflow** library that meets the following requirements:

1. **Load audio file** dataset **using** the **following requirements.**

Attributes	Description
Dataset Name	Tensorflow Speech Commands
Dataset Source Link	https://storage.googleapis.com/download.tensorflow.org/data/speech_commands_v0.01.tar.gz
Dataset Description	https://ml-exchange.org/datasets/tensorflow-speech-commands/

Figure 1. Dataset Requirements

2. **Divide the dataset** into **train**, **validation, and test sets** with the ratio of **80/10/10**.
3. **Perform two stages** of **preprocessing** on the **dataset**, which involves **squeezing** and **converting the audio array into a spectrogram** format that can be processed by the Convolutional Neural Network model.

4. Create the following **Convolutional Neural Network** architecture using **tensorflow** library which the **architecture modeling** is mainly divided into **3 parts**.

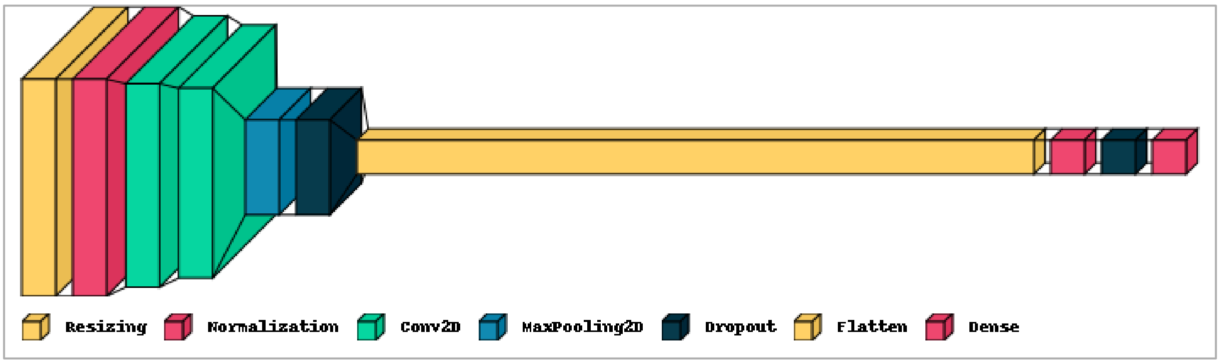
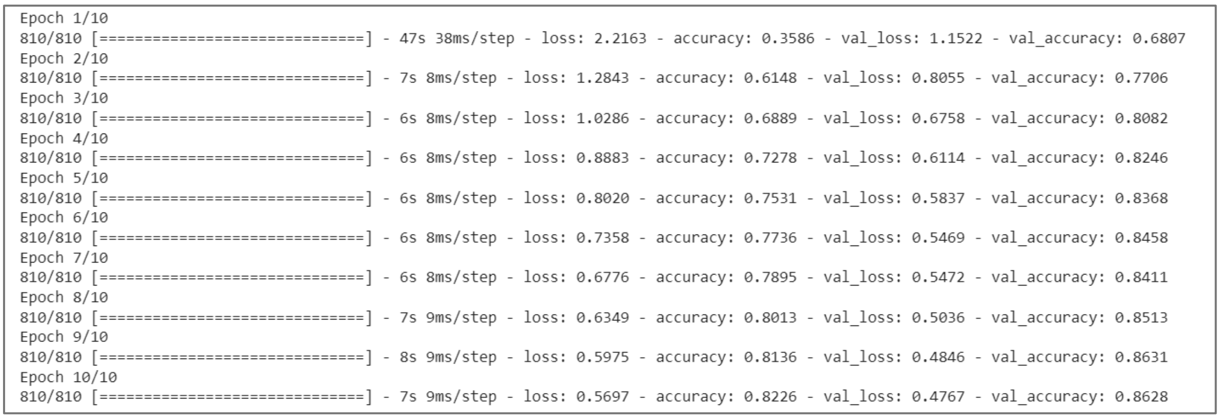


Figure 2. CNN Model Architecture

- **Input Layers**, defines the **input features** that is obtained from the spectogram preprocessing result.
- **Hidden Layers**, consists of the **main Convolutional Layer** which applies a set of filters to the input data, which helps to extract relevant features and patterns from the data.
- **Output Layers**, fully-connected layer to classify and generate the final output from patterns and features on the data.

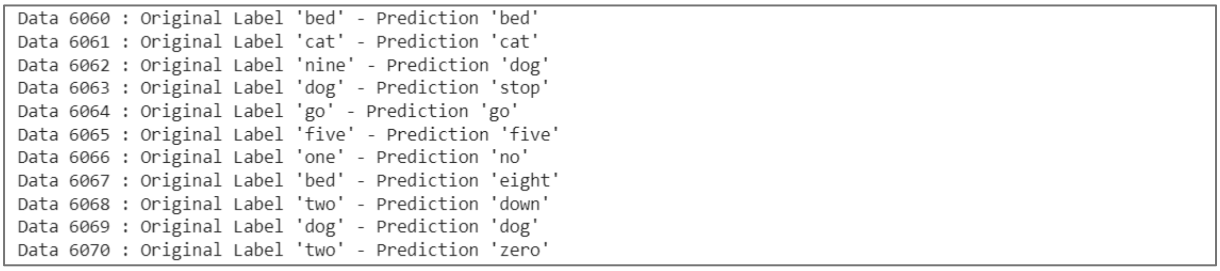
5. Train the **Convolutional Neural Network** model using the **train** and **validation set.**



```
Epoch 1/10
810/810 [==============================] - 47s 38ms/step - loss: 2.2163 - accuracy: 0.3586 - val_loss: 1.1522 - val_accuracy: 0.6807
Epoch 2/10
810/810 [==============================] - 7s 8ms/step - loss: 1.2843 - accuracy: 0.6148 - val_loss: 0.8055 - val_accuracy: 0.7706
Epoch 3/10
810/810 [==============================] - 6s 8ms/step - loss: 1.0286 - accuracy: 0.6889 - val_loss: 0.6758 - val_accuracy: 0.8082
Epoch 4/10
810/810 [==============================] - 6s 8ms/step - loss: 0.8883 - accuracy: 0.7278 - val_loss: 0.6114 - val_accuracy: 0.8246
Epoch 5/10
810/810 [==============================] - 6s 8ms/step - loss: 0.8020 - accuracy: 0.7531 - val_loss: 0.5837 - val_accuracy: 0.8368
Epoch 6/10
810/810 [==============================] - 6s 8ms/step - loss: 0.7358 - accuracy: 0.7736 - val_loss: 0.5469 - val_accuracy: 0.8458
Epoch 7/10
810/810 [==============================] - 6s 8ms/step - loss: 0.6776 - accuracy: 0.7895 - val_loss: 0.5472 - val_accuracy: 0.8411
Epoch 8/10
810/810 [==============================] - 7s 9ms/step - loss: 0.6349 - accuracy: 0.8013 - val_loss: 0.5036 - val_accuracy: 0.8513
Epoch 9/10
810/810 [==============================] - 8s 9ms/step - loss: 0.5975 - accuracy: 0.8136 - val_loss: 0.4846 - val_accuracy: 0.8631
Epoch 10/10
810/810 [==============================] - 7s 9ms/step - loss: 0.5697 - accuracy: 0.8226 - val_loss: 0.4767 - val_accuracy: 0.8628
```

Figure 3. Model Training

6. Make a **prediction** on a **test datasets** and **display** the **results** with the **highest probability.**



```
Data 6060 : Original Label 'bed' - Prediction 'bed'
Data 6061 : Original Label 'cat' - Prediction 'cat'
Data 6062 : Original Label 'nine' - Prediction 'dog'
Data 6063 : Original Label 'dog' - Prediction 'stop'
Data 6064 : Original Label 'go' - Prediction 'go'
Data 6065 : Original Label 'five' - Prediction 'five'
Data 6066 : Original Label 'one' - Prediction 'no'
Data 6067 : Original Label 'bed' - Prediction 'eight'
Data 6068 : Original Label 'two' - Prediction 'down'
Data 6069 : Original Label 'dog' - Prediction 'dog'
Data 6070 : Original Label 'two' - Prediction 'zero'
```

Figure 4. Model Prediction

7. Evaluate the **Convolutional Neural Network** model by **train accuracy** and **validation accuracy**.

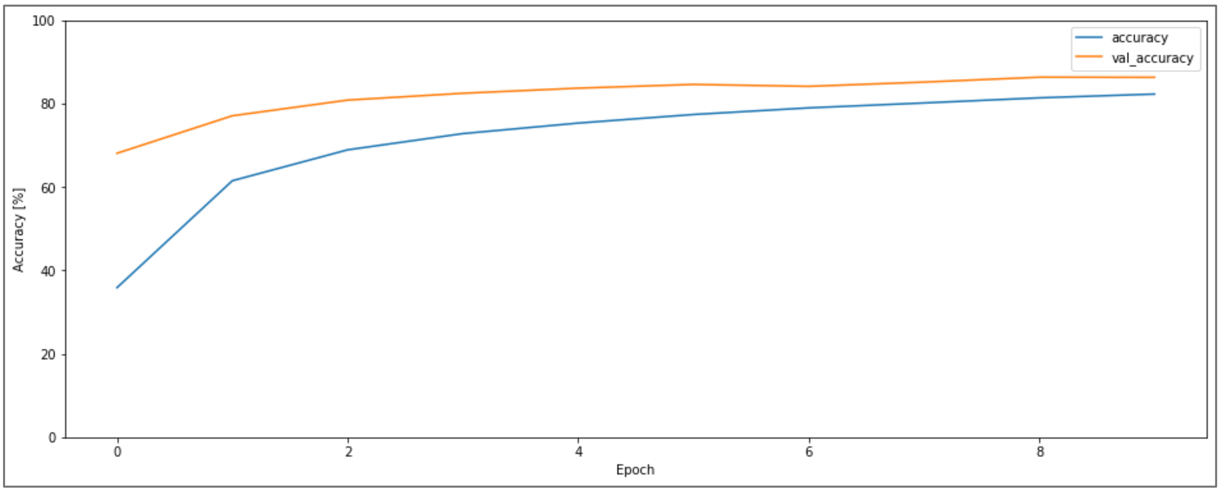


Figure 5. Model Evaluation