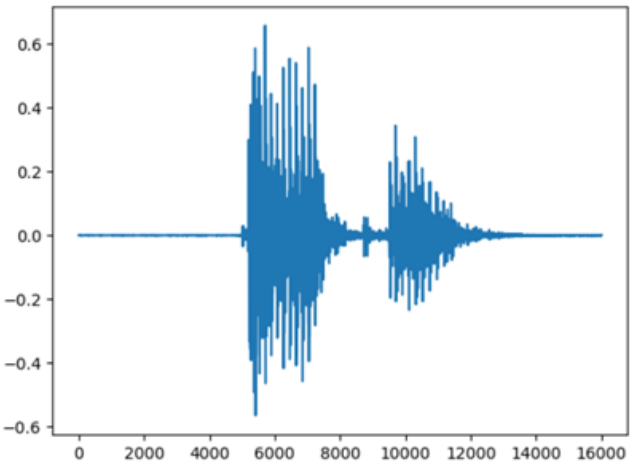
AUDIO CLASSIFICATION USING CNN

Alice Menna-502172

1. INTRODUCTION

The notebook is organized in different sections, each with a specific purpose. First, we upload both libraries,such as Pytorch or Matplotlib, and the dataset. The chosen dataset is the “SPEECHCOMMANDS” dataset, available on the basic torchaudio datasets. After separating the data in training and test we add some support functions to plot waveforms or play audio. Another important step is the preprocessing. Data preprocessing include the preparation of batches of audio tensors but, since waveforms have different lengths, padding sequences are performed. By padding sequences, we make all tensors in a batch of the same length by adding zeros. After preprocessing, we build the models and train them to evaluate their accuracy.



2. VERY DEEP CONVOLUTIONAL NETWORKS

In the M5 architecture, four convolutional layers are employed to extract spatial features from the input audio spectrograms. The first convolutional layer utilizes a kernel size of 80, followed by batch normalization and max pooling. The second, the third and fourth convolutional layer applies a smaller kernel size of 3, followed by batch normalization and max pooling. A fully connected layer is utilized to map the extracted features and classify them into the categories. The activation functions

used in the model are the Rectified Linear Unit (ReLU) activation function and the Softmax Activation function to assign probabilities to the categories.

In the M3 architecture, two convolutional layers are employed to extract spatial features from the input audio spectrograms. The first convolutional layer utilizes a kernel size of 80, to extract low-level features, followed by batch normalization and max pooling. The second convolutional layer applies a smaller kernel size of 3 with padding for more localized features, followed by batch normalization and max pooling. A fully connected layer is utilized to map the extracted features and classify them into the categories. The activation functions used in the model are the Rectified Linear Unit (ReLU) activation function and the Softmax Activation function to assign probabilities to the categories.

```
class M3(nn.Module):
    def __init__(self, n_input=1, n_output=35, stride=4, n_channel=128):
        super().__init__()
        self.conv1 = nn.Conv1d(n_input, 2*n_channel, kernel_size=80, stride=stride)
        self.bn1 = nn.BatchNorm1d(2*n_channel)
        self.pool1 = nn.MaxPool1d(4)
        self.conv2 = nn.Conv1d(2*n_channel, 2*n_channel, kernel_size=3, stride=1, padding=1)
        self.bn2 = nn.BatchNorm1d(2*n_channel)
        self.pool2 = nn.MaxPool1d(4)
        self.fc1 = nn.Linear(2 * n_channel, n_output)

    def forward(self, x):
        x = self.conv1(x)
        x = F.relu(self.bn1(x))
        x = self.pool1(x)
        x = self.conv2(x)
        x = F.relu(self.bn2(x))
        x = self.pool2(x)
        x = F.avg_pool1d(x, x.shape[-1])
        x = x.permute(0, 2, 1)
        x = self.fc1(x)
        return F.log_softmax(x, dim=2)
```

We train both models using Adam optimiser for 3 epochs.

3. RESULTS AND ANALYSES

Table1 shows the accuracy and the execution time of both M3 and M5 architecture testing on 3 epochs with a batch size of 256. Compared to M5, M3 performs very poorly. This indicates that 2 layers are not sufficient to perform an accurate audio recognition.

Model	Accuracy	Execution Time
M3	2%	19 min
M5	74%	17 min

Table 2 shows the results of the confusion matrix of the M5 architecture.

```
Training Set:
True Positives (TP): 8173
False Positives (FP): 6039
True Negatives (TN): 6496
False Negatives (FN): 6259

Test Set:
True Positives (TP): 21191
False Positives (FP): 1292
True Negatives (TN): 11272
False Negatives (FN): 1404
```

Although M5 reach an accuracy of 74%, there are some ways to improve it:

- **Batch Size**

Reducing batch size can increase the accuracy of the model. In particular, 3 different batch size have been proposed: 128,400 in addition to the original of 256. The results show that a batch size of 128 produces an accuracy of 78%.

- **Epochs**

Increasing the number of epochs can lead to an improvement of the accuracy. The results show that using 11 epochs produces an accuracy of 80%.

- **M11 Architecture**

In M11 architecture there are 11 convolutional layers. Having a higher number of convolutional layers increases the accuracy of the model since it extracts features in a deeper way.

4. CONCLUSIONS

In this work, we analyze the performance of a deep convolutional network that operates directly on acoustic wave inputs. The results show that a deep convolutional network with 4 or 10 layers performs better that a 2-layer CNN, reaching a final accuracy of 80% considering 11 epochs and a batch size of 256.