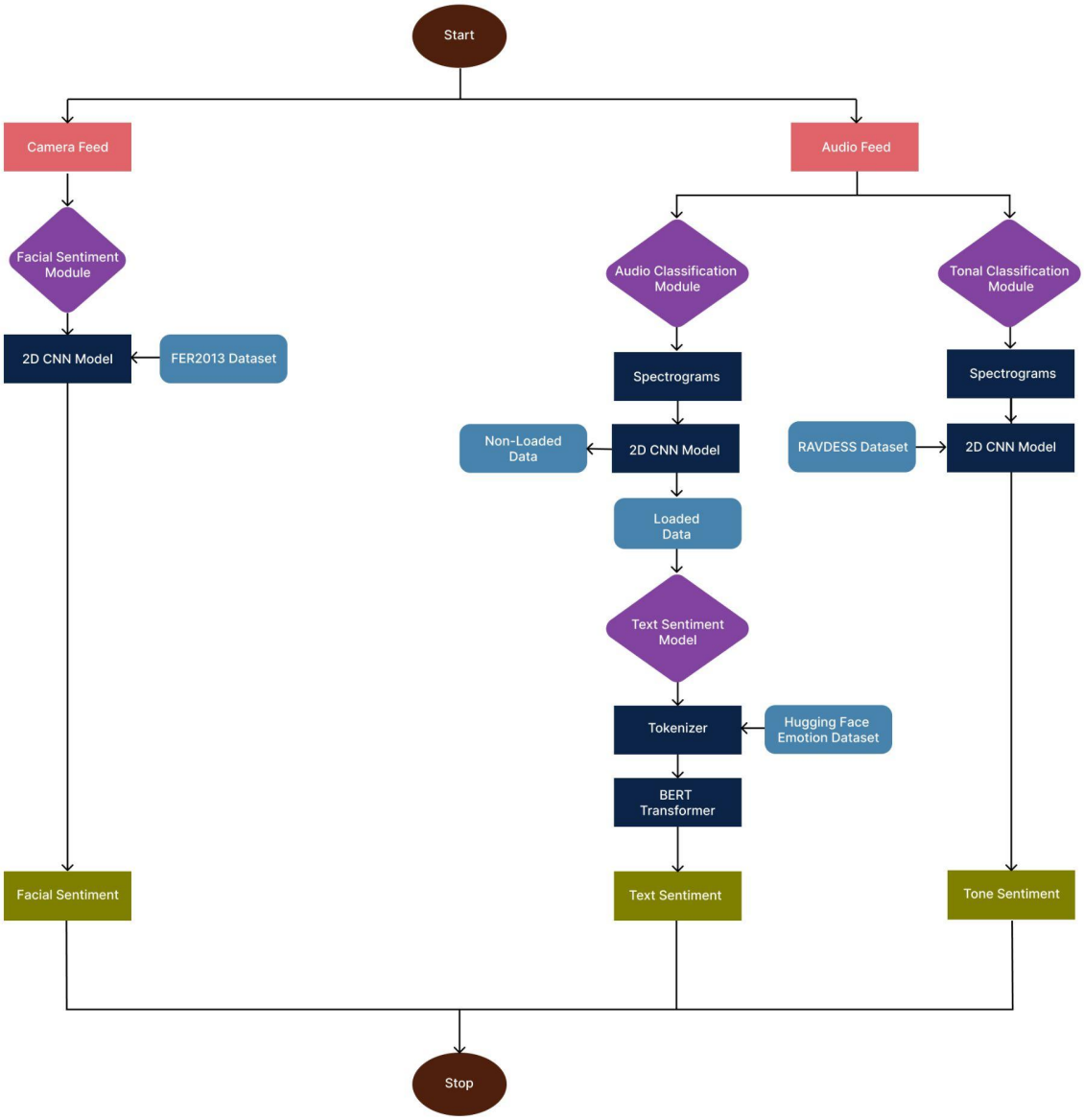
Opinion Mining of Audio Using Machine Learning Techniques
(Project Analysis)

System Design



Creating the Audio Classification Model

- Dataset:

	Initial Data	After Augmenting
Loaded	70	121
Non-Loaded	100	129

- Algorithm:

```
# Defining the model
SequentialModel (Convolution2D, Pooling, Dense, Activation)
    Input image shape for different convolutions
    Add 2D convolutions with 32 filters and size 3*3 until filter reaches 64
    Perform Sigmoid and ReLU activations till dense reached 128 layers
    Involve mask of size 2*2 through max pooling operations until the dense layer is not reached
    Hyper-tune model with drop rate of 0.5 to avoid overfitting
    Add fully connected layer with Dense classes
    Perform Softmax activation through Adam optimizer to avoid model decay
    Define the accuracy metric and compile the model with binary_crossentropy as loss function

# Training
    Perform model fitting using batch size and epochs
    Validation of data also tested till training completed

# Visualizing losses and accuracy
    Visualize training loss and validation loss with obtained accuracy until 10 epochs

# Testing the Model
    Test the model against the test samples and plot the confusion matrix with appropriate metrics
```

- Feature Extraction to train a 1D CNN model:

```
print(features, '\n', labels)

[[-4.13202148e+02  9.98437042e+01  8.94720840e+00 ... -1.55165105e-04
  -1.73239641e-02 -1.01103534e-02]
 [-3.78620789e+02  1.70048584e+02 -1.47591953e+01 ... -7.65202379e-03
  -5.87135588e-04 -1.63093681e-02]
 [-2.93391785e+02  1.51812531e+02 -2.80752163e+01 ... -1.10498545e-02
  -6.49728064e-03 -5.60485430e-03]
 ...
 [-6.06264832e+02  6.31914139e+01 -1.19954929e+01 ...  5.64947715e-03
   2.24111090e-02 -1.09439897e-02]
 [-6.36193115e+02  6.22383614e+01 -3.28203726e+00 ...  1.95569443e-02
   1.15436315e-02 -2.71684398e-03]
 [-3.82944397e+02  5.38968201e+01 -2.32973309e+01 ...  3.52986184e-04
   1.80919608e-02 -4.88506067e-03]]
 [1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2]
```

The pyaudioclassification library extracts 5 features from each audio file namely: **MFCC, Chroma, Mel, Contrast, and Tonnetz**. All the values generated are then concatenated into one list. This list has a total of 193 feature values for each of the audio files where loaded files are represented with label 1 and non-load with label 2.

- Model Summary (1D CNN):

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv1d (Conv1D)             (None, 191, 64)           256

 activation (Activation)     (None, 191, 64)           0

 conv1d_1 (Conv1D)           (None, 189, 64)           12352

 activation_1 (Activation)   (None, 189, 64)           0

 max_pooling1d (MaxPooling1D (None, 63, 64)            0
 )

 conv1d_2 (Conv1D)           (None, 61, 128)           24704

 max_pooling1d_1 (MaxPooling (None, 20, 128)           0
 1D)
```

```
conv1d_3 (Conv1D)           (None, 18, 128)           49280

global_average_pooling1d    (None, 128)               0
(GlobalAveragePooling1D)

dropout (Dropout)           (None, 128)               0

dense (Dense)               (None, 1)                 129

=================================================================
Total params: 86,721
Trainable params: 86,721
Non-trainable params: 0
```

- We are using the Sequential() model, a linear stack of layers i.e., the output of one layer will be given to the next in a sequential way. That's why in the model summary we get, sequential as the name. We could have different kinds, 'Dense', 'RNN', 'LSTM', etc.

- The input_shape parameter in the CONV1D function takes 3 values. **(batch_size, time_steps, input_dims)**
  We have given as (193, 1). This means **batch_size will be set to None** so that we can specify any value in the train function, time_steps will be 193 and the number of feature dimensions is 1 because we have provided only one feature array.

- Adam is often considered to be a more robust and efficient optimization algorithm compared to SGD. Adam is less sensitive to the choice of hyperparameters such as the learning rate and momentum, and can often converge faster and more reliably than SGD. We have used SGD as it converges slowly to see the training process.

- So, our input shape is of the form (None, 193, 1). If we see the Conv1D function, we have chosen 64 filters of kernel size 3 to be convolved on the input.
  The output shape is **(batch_size, time_steps, channels, or filters).** Batch_size is None as discussed earlier and time_steps is calculated as:
  input_shape[0] - kernel_size + 1 = 193 - 3 + 1 = 191

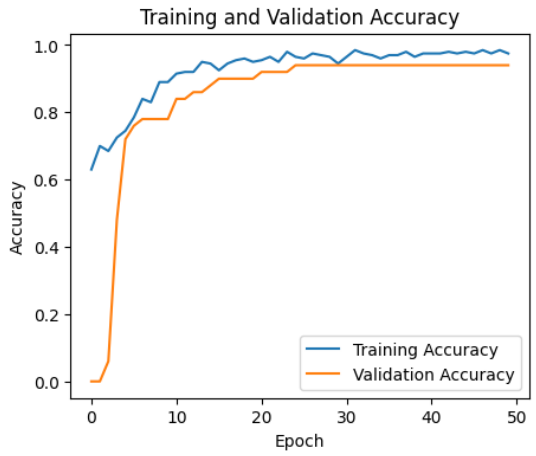
- Parameters are calculated as: ***kernel_size * input_channels * filters + bias***
  = 3 * 1 * 64 + 64 = 256
  Similarly, for the next layer, 3 * 64 * 64 + 64 = 12352

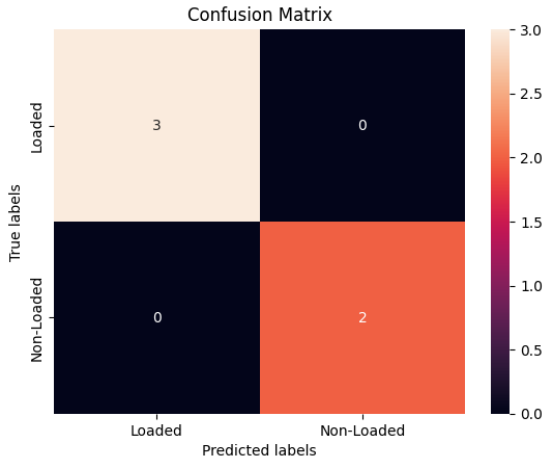
- Total Trainable parameters are the total number of weights and biases used in between the neural network which is nothing but the sum of the parameters shown in the model summary.

- Non-Trainable parameters can be set like:
  model.layers[0].trainable = False
  We haven't set any so it's 0.

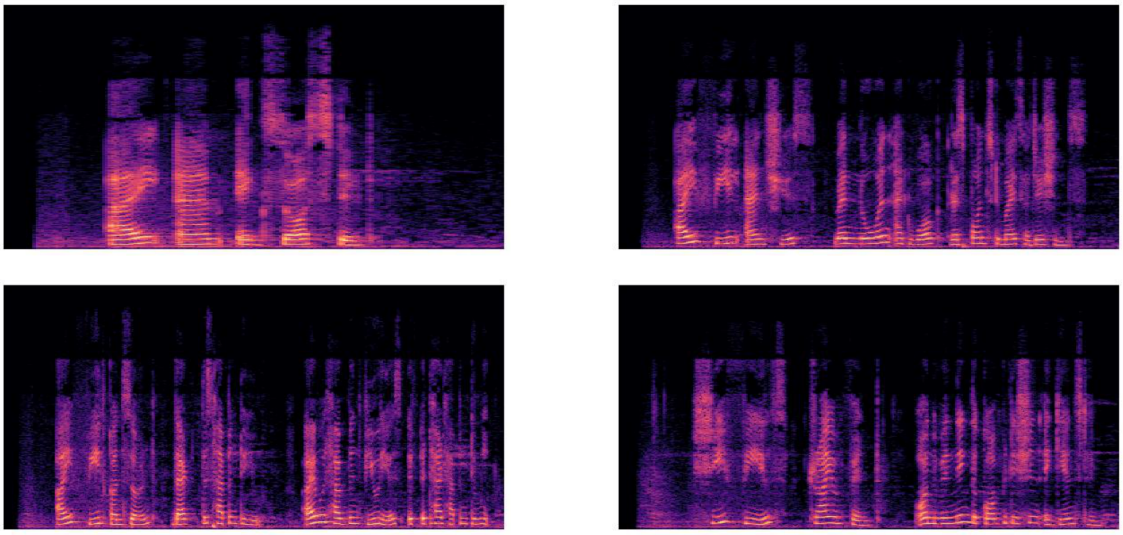
- The value below EPOCH is the number of batches the training occurs for each epoch. It is calculated as ***total_no_of_training_samples / batch_size***
  For our example, it comes out to be 250 / 10 = 25. But the number shown is 20 because 20 percent of the data sample is taken as the validation set. So 20 batches are taken for training and 5 batches are taken for validation.

- The model summary table name is '**sequential_(no. Of times the cell was run)**'
- Model Evaluation (1D CNN):

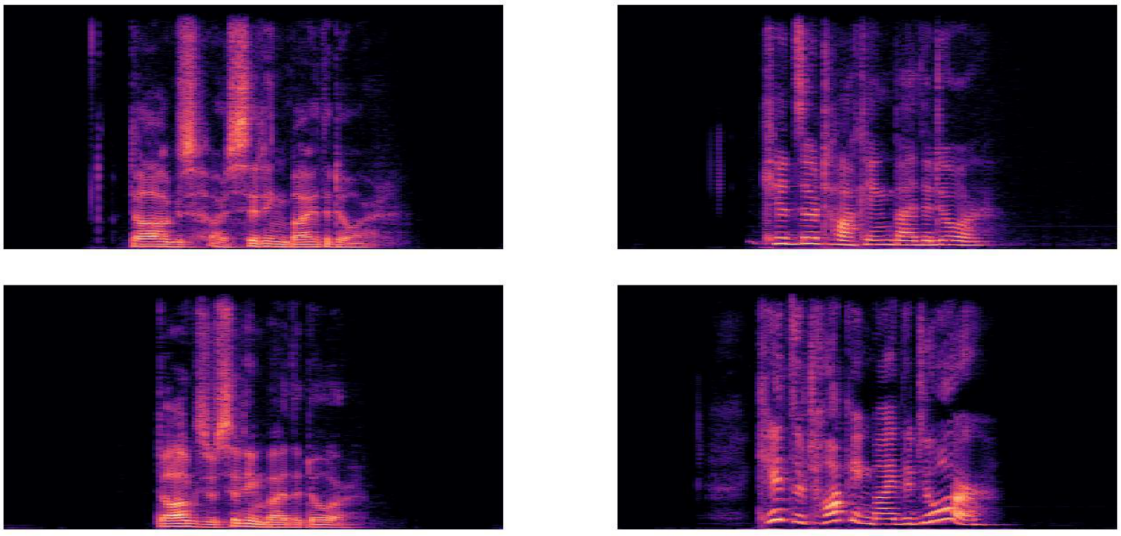




- Feature Extraction to train a 2D CNN model:



Spectrogram Samples for Loaded audio

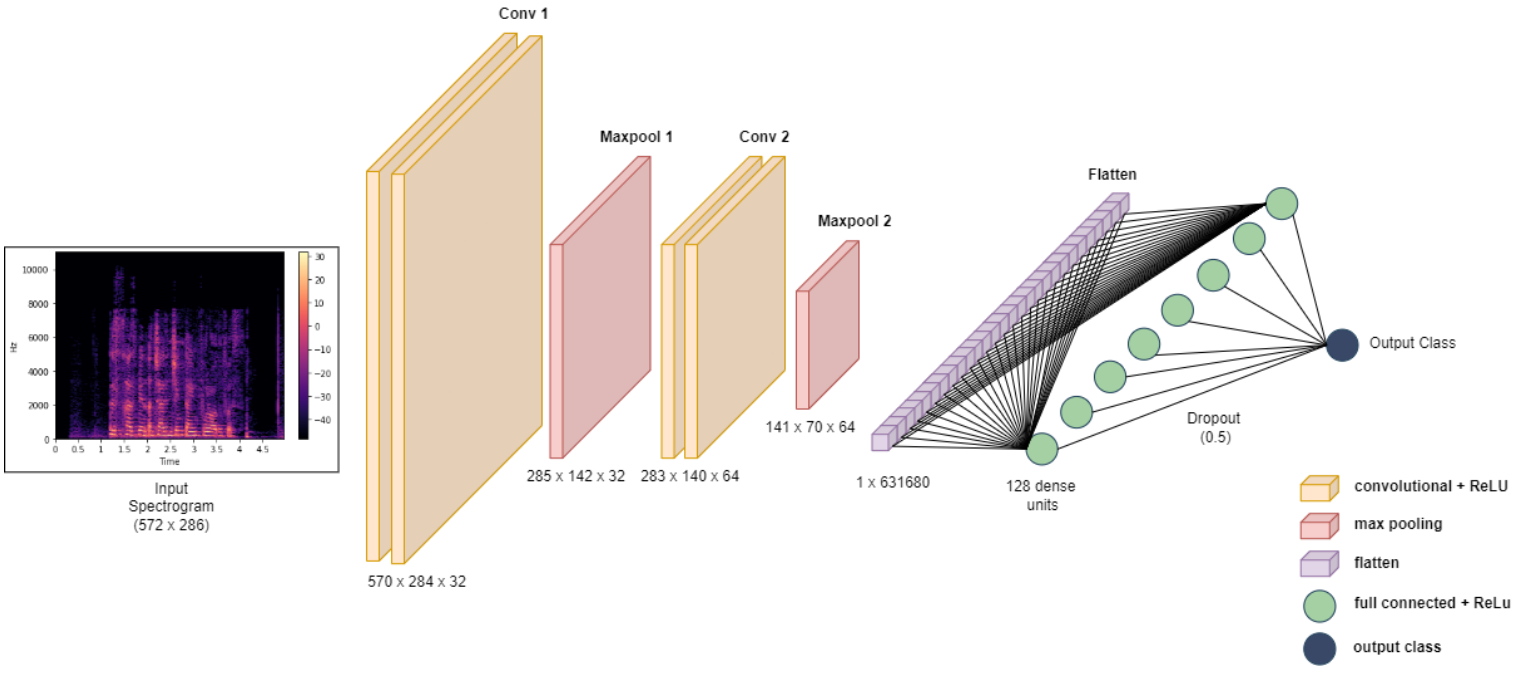


Spectrogram Samples for Non-Loaded audio

- Model Summary (2D CNN):

```
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                 Output Shape              Param #
=================================================================
 conv2d (Conv2D)              (None, 570, 284, 32)      896

 max_pooling2d (MaxPooling2D  (None, 285, 142, 32)      0
 )

 conv2d_1 (Conv2D)            (None, 283, 140, 64)      18496

 max_pooling2d_1 (MaxPooling  (None, 141, 70, 64)       0
 2D)

 flatten (Flatten)            (None, 631680)            0

 dense_1 (Dense)              (None, 128)               80855168

 dropout_1 (Dropout)          (None, 128)               0

 dense_2 (Dense)              (None, 1)                 129

=================================================================
Total params: 80,874,689
Trainable params: 80,874,689
Non-trainable params: 0
```

- Model Architecture (2D CNN):



- We are using the Sequential() model, which represents a linear stack of layers in a sequential manner. In this model, the output of one layer is passed as input to the next layer. The model summary shows the architecture of the 2D CNN.

- The first layer in the model is a Conv2D layer. It performs a 2-dimensional convolution on the input data. The layer has **32 filters with a kernel size of (570, 284).** The number of filters determines the number of feature maps generated. The larger the kernel size, the larger the receptive field of the filters. Therefore, each filter will scan a window of size 570x284 over the input. The output of this layer has a shape of **(None, 570, 284, 32),** indicating that the batch size is not fixed, and there are 32 feature maps generated.

- Following the Conv2D layer, we have a MaxPooling2D layer. This layer performs 2D max pooling, reducing the spatial dimensions of the previous layer's output by taking the maximum value within each pooling window. The pooling window size is (2, 2), which means that it reduces the **spatial dimensions by a factor of 2** in both width and height. In this case, the output shape becomes **(None, 285, 142, 32),** where the width and height are halved while the number of channels remains the same.

- The next layer is another Conv2D layer with 64 filters and a kernel size of (283, 140). It convolves these filters over the previous layer's output and produces feature maps of size (None, 283, 140, 64). Increasing the number of filters allows the layer to capture more diverse patterns and increase the complexity of the model.

- After the second Conv2D layer, we have another MaxPooling2D layer with a pooling window size of (2, 2). Similar to the previous pooling layer, it reduces the spatial dimensions by a factor of 2 in both width and height. This results in an output shape of (None, 141, 70, 64).

- The Flatten layer reshapes the 4-dimensional output into a 2-dimensional tensor of shape (None, 631680), which is necessary to connect with the following Dense layers. The total number of elements in the flattened tensor is the product of the dimensions, which in this case is **141 * 70 * 64 = 631680.**

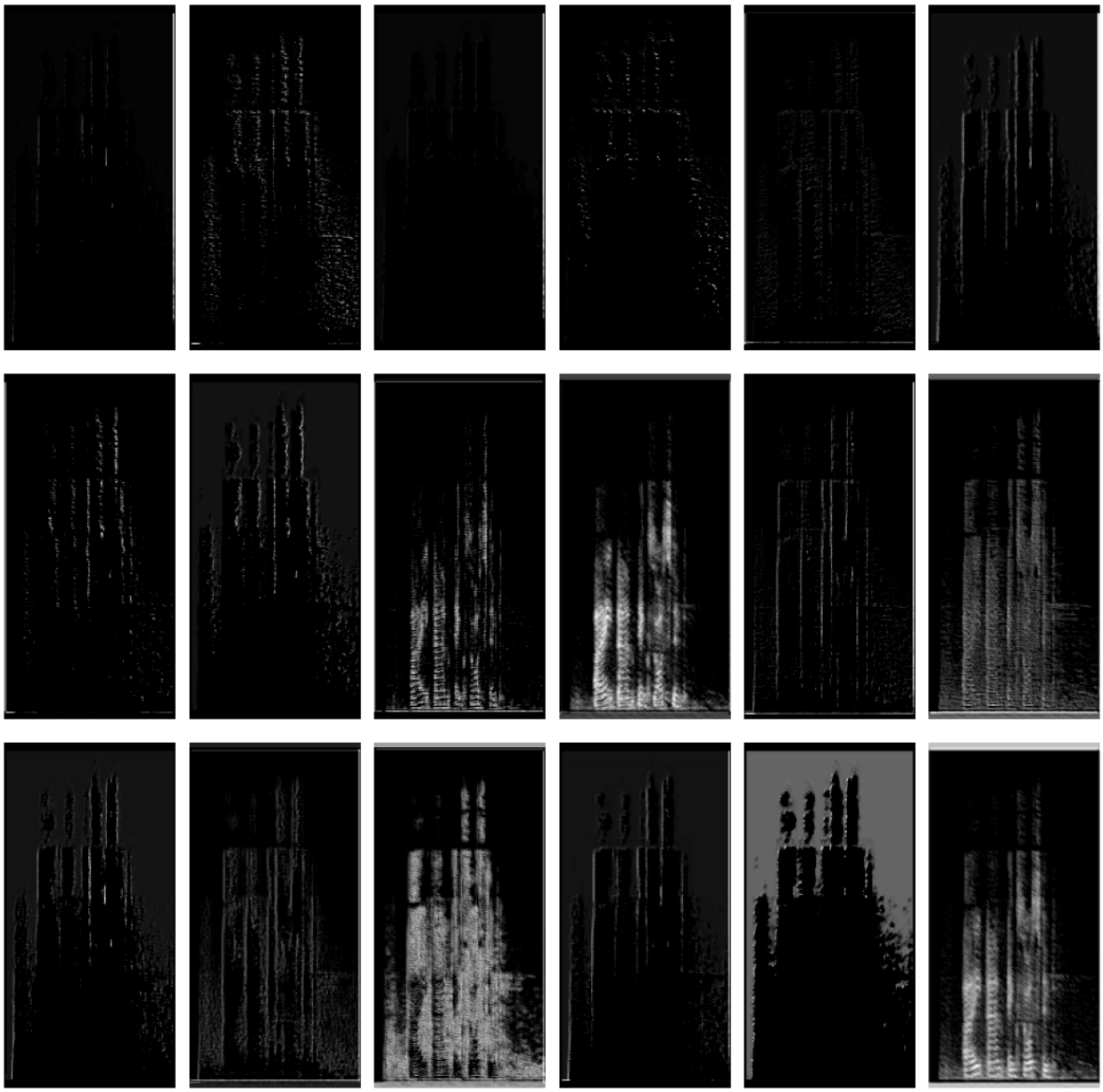
- Next, we have a Dense layer with 128 units. It takes the flattened input and applies a linear transformation with 128 learnable weights. The output shape of this layer is (None, 128). The number of units in this layer determines the dimensionality of the output space.

- To prevent overfitting and improve generalization, a Dropout layer with a rate of 0.5 is added. This layer randomly sets a fraction of input units to 0 during training, reducing the interdependence between neurons and helping to prevent overfitting.

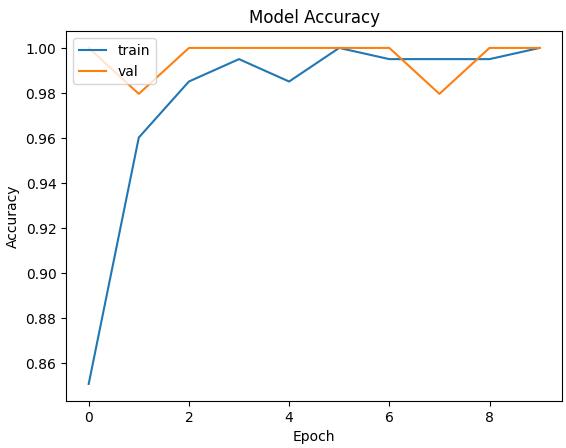
- Finally, we have the last Dense layer with a single unit, which represents the output of the model. It produces a single scalar value as the model's prediction.
  The total trainable parameters in the model are calculated by summing up the number of weights and biases in all the layers. In this case, it is **80,855,298.** The number of parameters in each layer depends on the ***size of the kernel/filter, the number of input channels, the number of filters/output channels, and the presence of biases.***

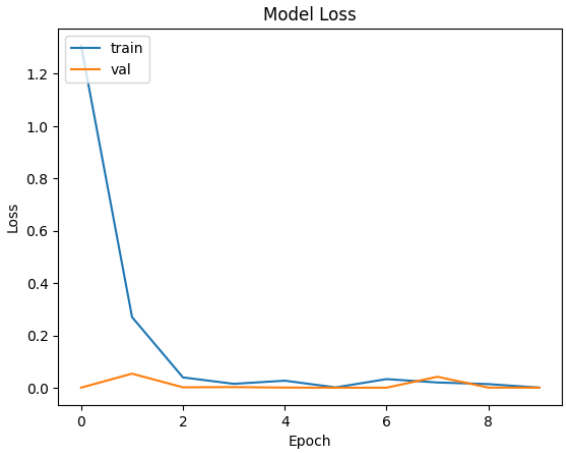
- Feature maps generated by intermediate layer of 2D CNN:

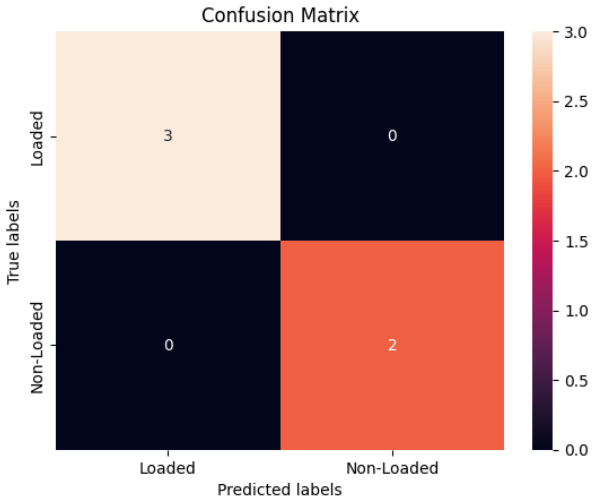
feature maps - convlayer2



- Model Evaluation (2D CNN):

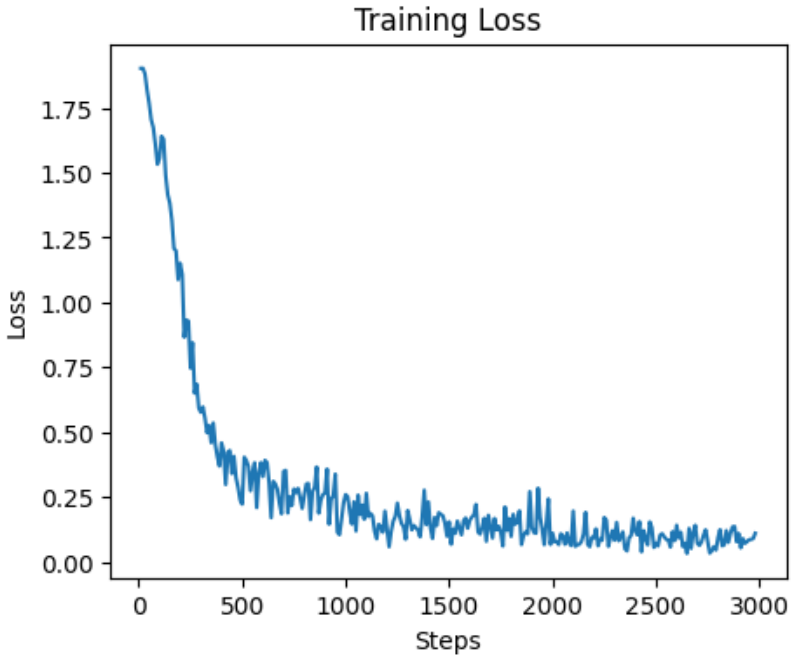


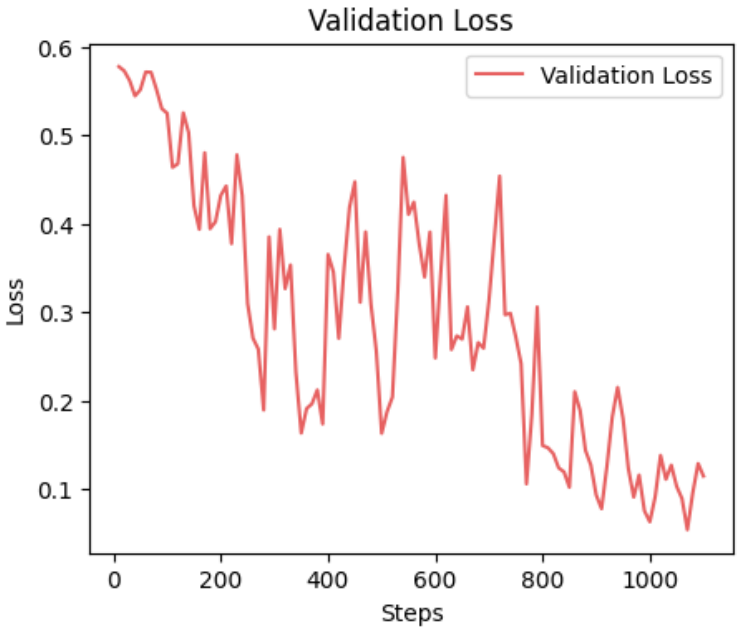


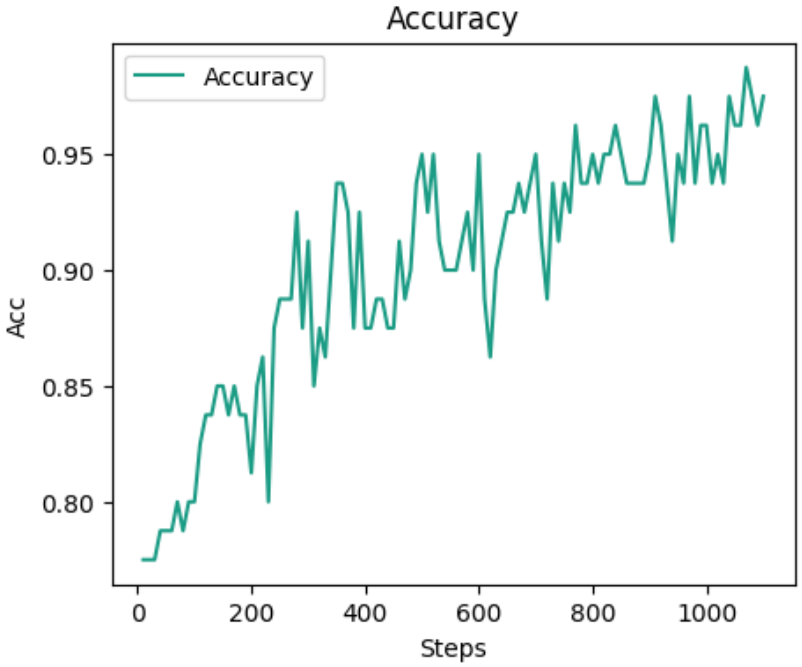


Creating the Text Classification Model

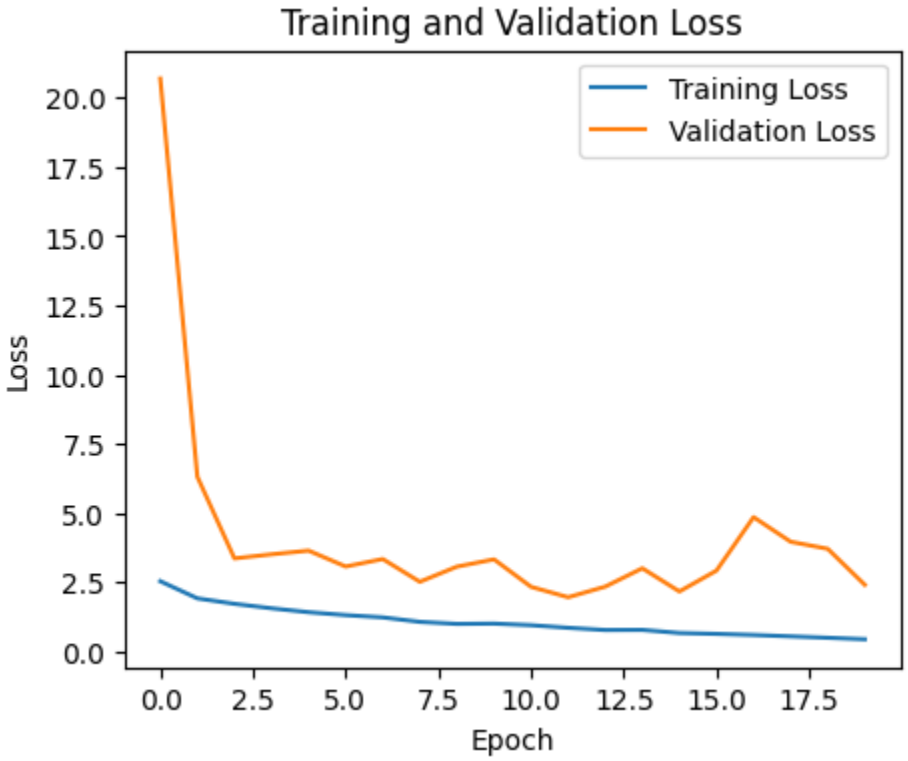
- Model Evaluation

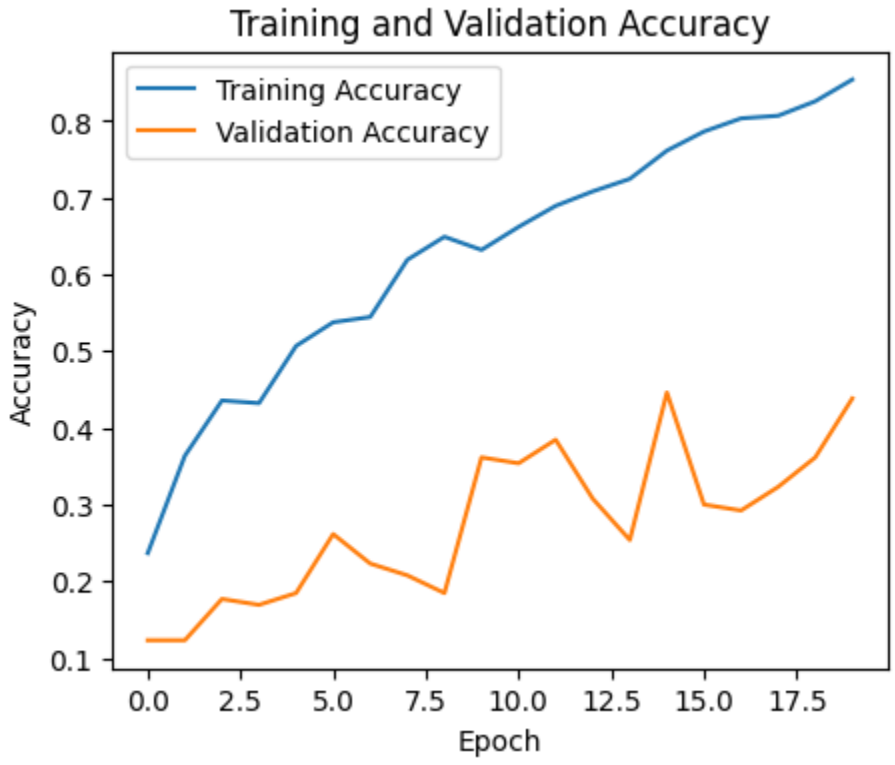



Validation Loss
Validation Loss
Loss
Steps
0.6
0.5
0.4
0.3
0.2
0.1
0
200
400
600
800
1000

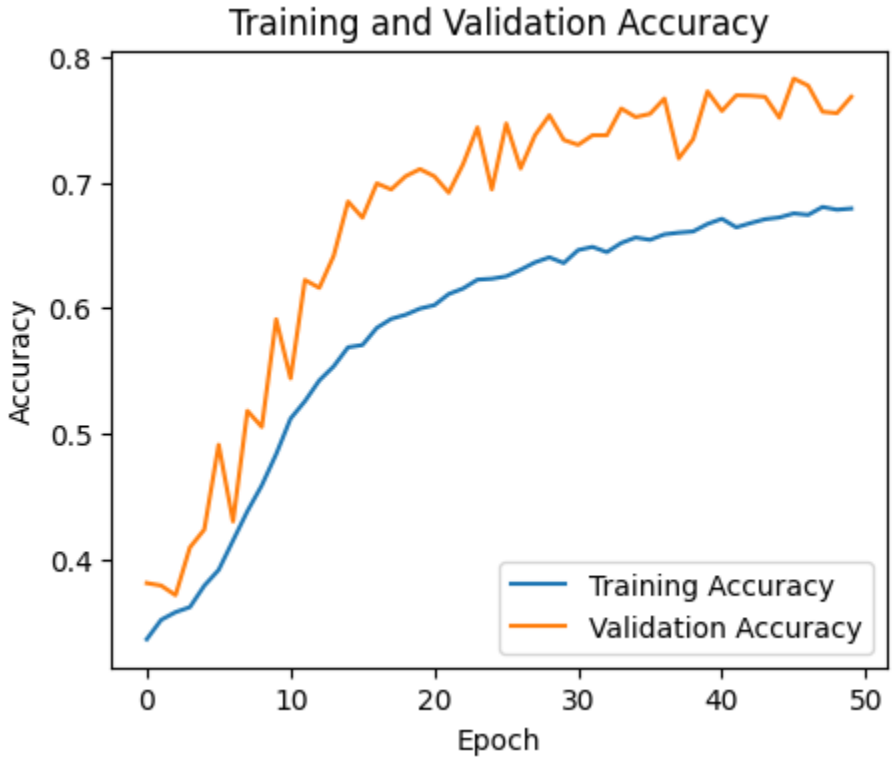

Accuracy
Accuracy
Acc
Steps
0.95
0.90
0.85
0.80
0
200
400
600
800
1000

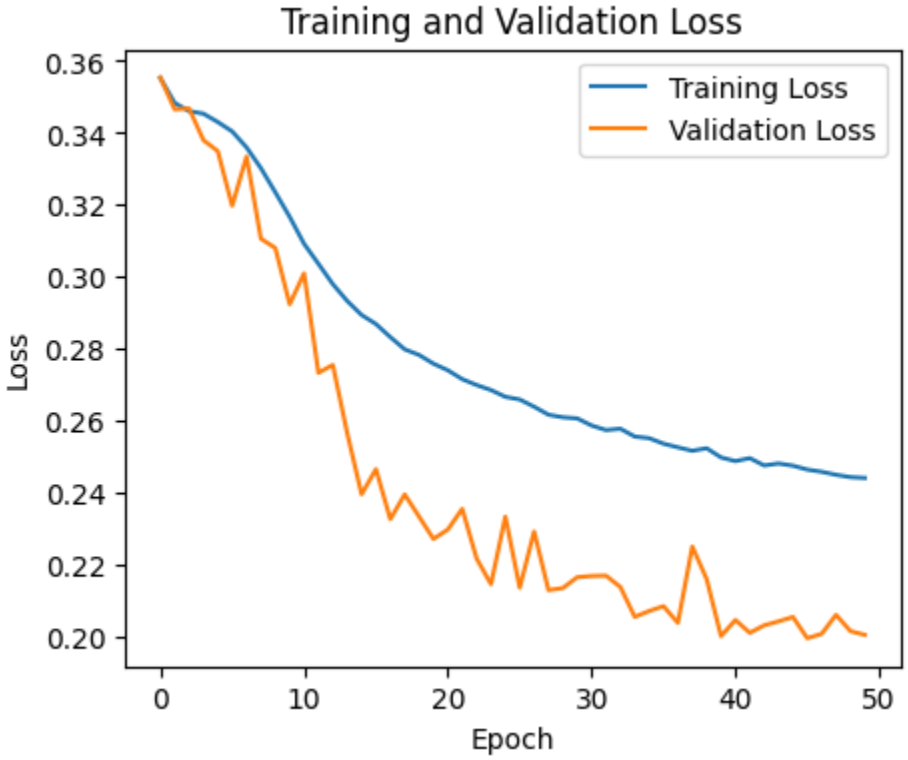
Creating the Tonal Classification Model

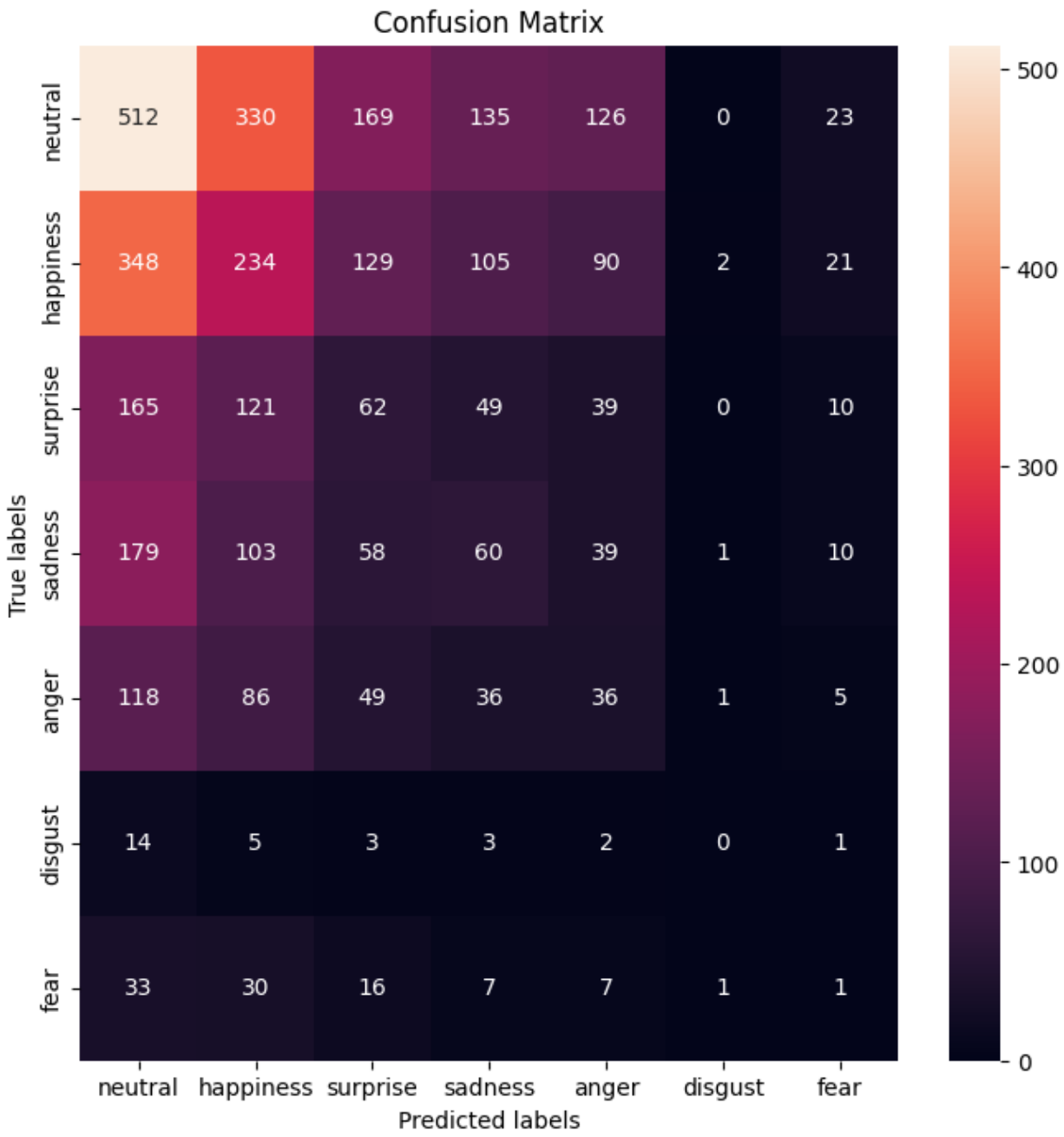




Creating Face Sentiment Analysis Model






Confusion Matrix
True labels
neutral
happiness
surprise
sadness
anger
disgust
fear
512 330 169 135 126 0 23
348 234 129 105 90 2 21
165 121 62 49 39 0 10
179 103 58 60 39 1 10
118 86 49 36 36 1 5
14 5 3 3 2 0 1
33 30 16 7 7 1 1
neutral
happiness
surprise
sadness
anger
disgust
fear
Predicted labels
500
400
300
200
100
0

Models	Accuracy	Training Time
Audio Classification	98	15 min 10 sec
Text Sentiment Analysis	97	1 hr 12 min 20 sec
Face Sentiment Analysis	78	1 hr 0 min 11 sec
Tonal Sentiment Analysis	64	30 min 47 sec