IT353 : Deep Learning

Lab Assignment 3 - Audio and Text Classification

Kesanam Ashinee

211AI023

Dataset 1: (For Audio Classification

The dataset describes 440 recordings of cat meows categorized into three different situations:

- Waiting for food: This category likely contains recordings of meows associated with hunger or anticipation of receiving food.
- Isolation in unfamiliar environment: This category likely includes meows associated with stress or anxiety from being in an unfamiliar place.
- Brushing (being brushed affectionately by the owner): This category likely contains recordings of meows associated with positive interactions or contentment during petting.

Additional details:

- The dataset involves 21 cats belonging to two breeds: Maine Coon and European Shorthair.
- The purpose of the dataset is potentially to contribute to understanding cat vocalizations and their correlation with specific contexts or emotions.

Link: Cat Meow Classification Dataset

Dataset 2: (For Text Classification)

This dataset is designed for classifying e-commerce product descriptions into four categories:

- Electronics: Products related to electronic devices and accessories.
- Household: Products related to household items and appliances.
- Books: Products related to books and publications.
- Clothing & Accessories: Products related to clothing, footwear, and accessories.

The dataset contains two columns:

- Class Name: This column specifies the category of the product description, represented as a string ("Electronics", "Household", "Books", or "Clothing & Accessories").
- Product Description: This column contains the textual description of the product, providing information about its features, specifications, and other relevant details.

Dataset Link: E-Commerce Text Classification Dataset

Kaggle Notebook

All code can be found here - Task 1 Notebook , Task 2 Notebook

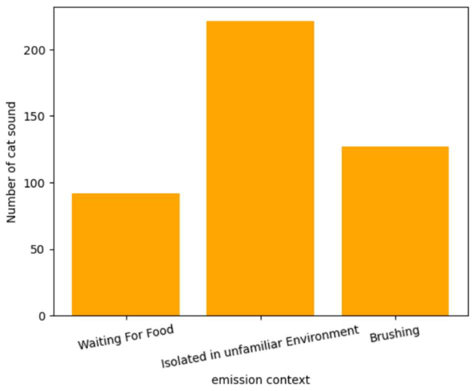
Task 1:

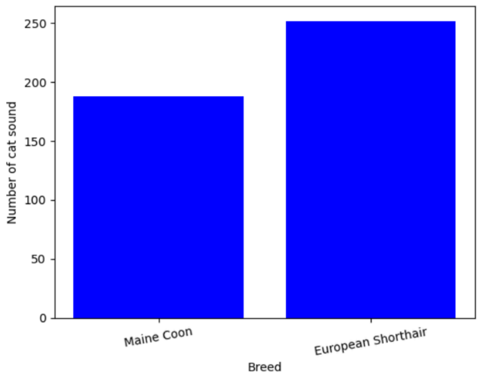
Download any Audio Classification dataset (Birds sound, Gender, vechicle sound etc) and convert it into spectrogram, use Deep CNN models and classify the spectrograms.

Implementation:

EDA:

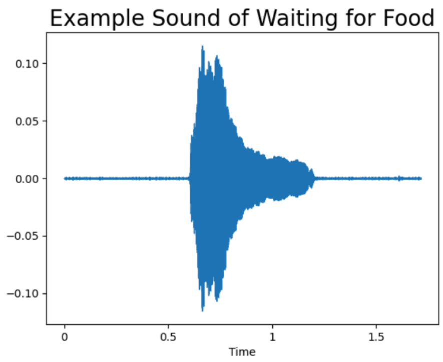
Checking the value counts for each category

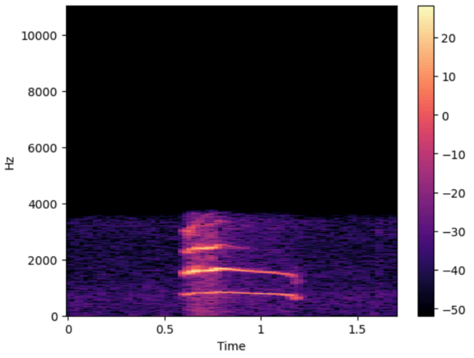


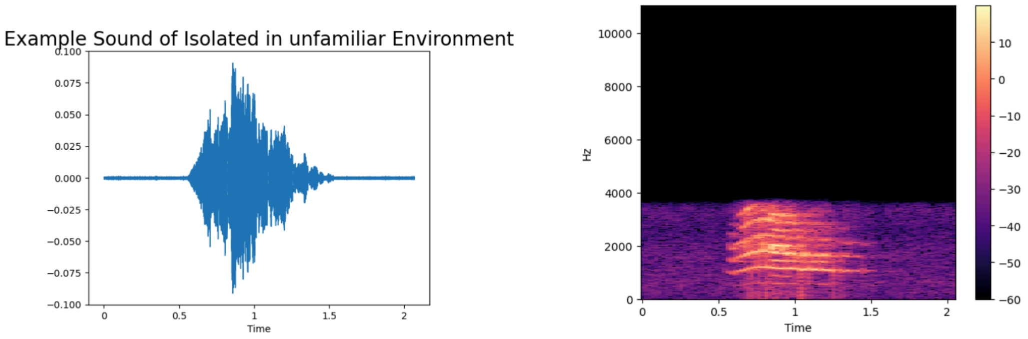


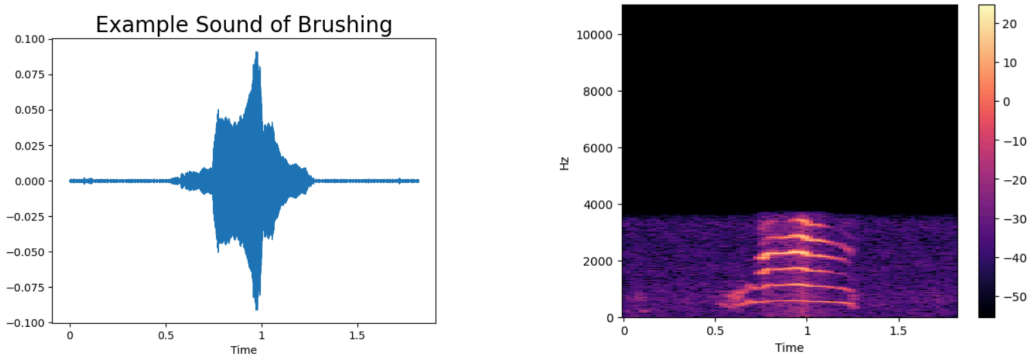
Data Loading and Visualization:

- The code loads audio files using librosa and visualizes the sound waveforms and spectrograms.
- The 3 types of meow are loaded and displayed.


Example Sound of Waiting for Food







Data Augmentation Functions:

- Functions are defined for data augmentation:
  - Adding noise to the audio data.
  - Stretching the audio data in time.
  - Pitch shifting the audio data.

Feature Extraction:

- Features are extracted from the audio data to be used for classification.
- Features include Zero Crossing Rate (ZCR), Chroma_stft, Mel-frequency cepstral coefficients (MFCC), Root Mean Square (RMS) value, and MelSpectogram.

Feature Extraction Function:

- The extract_features() function computes various features from the audio data.

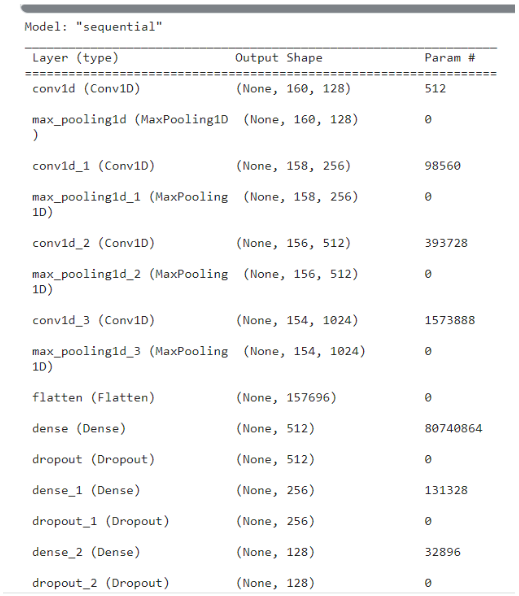
Feature Extraction from Augmented Data:

- Features are extracted from original data, noise-added data, and stretched/pitched data.

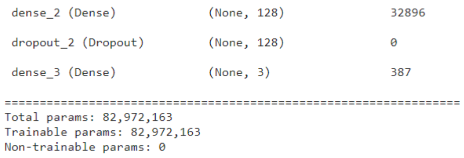
Model 1

Model Architecture:

- The model is a sequential model defined using Keras's Sequential API.
- It comprises several layers including Conv1D, MaxPooling1D, Flatten, and Dense layers.



```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #   
=================================================================
 conv1d (Conv1D)             (None, 160, 128)          512       
                                                                 
 max_pooling1d (MaxPooling1D  (None, 160, 128)         0         
 )                                                               
                                                                 
 conv1d_1 (Conv1D)           (None, 158, 256)          98560     
                                                                 
 max_pooling1d_1 (MaxPooling  (None, 158, 256)         0         
 1D)                                                             
                                                                 
 conv1d_2 (Conv1D)           (None, 156, 512)          393728    
                                                                 
 max_pooling1d_2 (MaxPooling  (None, 156, 512)         0         
 1D)                                                             
                                                                 
 conv1d_3 (Conv1D)           (None, 154, 1024)         1573888   
                                                                 
 max_pooling1d_3 (MaxPooling  (None, 154, 1024)        0         
 1D)                                                             
                                                                 
 flatten (Flatten)           (None, 157696)            0         
                                                                 
 dense (Dense)               (None, 512)               80740864  
                                                                 
 dropout (Dropout)           (None, 512)               0         
                                                                 
 dense_1 (Dense)             (None, 256)               131328    
                                                                 
 dropout_1 (Dropout)         (None, 256)               0         
                                                                 
 dense_2 (Dense)             (None, 128)               32896     
                                                                 
 dropout_2 (Dropout)         (None, 128)               0         
```



```
 dense_2 (Dense)             (None, 128)               32896     
                                                                 
 dropout_2 (Dropout)         (None, 128)               0         
                                                                 
 dense_3 (Dense)             (None, 3)                 387       
                                                                 
=================================================================
Total params: 82,972,163
Trainable params: 82,972,163
Non-trainable params: 0
```

Convolutional Layers:

- Four Conv1D layers with increasing number of filters (128, 256, 512, and 1024 respectively).
- Each Conv1D layer uses a kernel size of 3 and ReLU activation function.
- Convolutional layers are designed to capture hierarchical patterns in the input data.

MaxPooling Layers:

- Four MaxPooling1D layers with pool size (1).
- MaxPooling layers reduce the spatial dimensions of the input, helping in feature extraction and reducing computational complexity.

Flatten Layer:

- After the convolutional layers, there's a Flatten layer to flatten the 1D output from the convolutional layers into a 1D vector.

Dense Layers:

- Three Dense layers follow the Flatten layer with decreasing number of units (512, 256, and 128 respectively).
- Each Dense layer uses ReLU activation function.
- Dropout layers with a dropout rate of 0.3 are added after each Dense layer to prevent overfitting by randomly dropping a fraction of the input units.

Output Layer:

- The final Dense layer consists of 3 units with softmax activation, representing the probabilities of the input belonging to each of the three classes ('Waiting For Food', 'Isolated in unfamiliar Environment', 'Brushing').

Model Compilation:

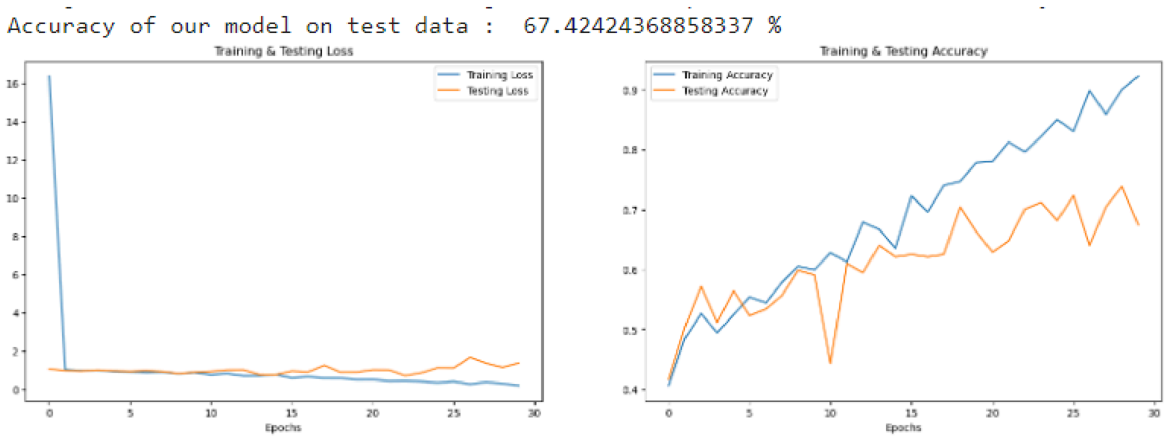
- The model is compiled using categorical cross-entropy loss function, RMSprop optimizer, and accuracy as the evaluation metric.

Model Training:

- The model is trained for 30 epochs with a batch size of 128.
- Training data and validation data are passed during the training process.

Model Evaluation:

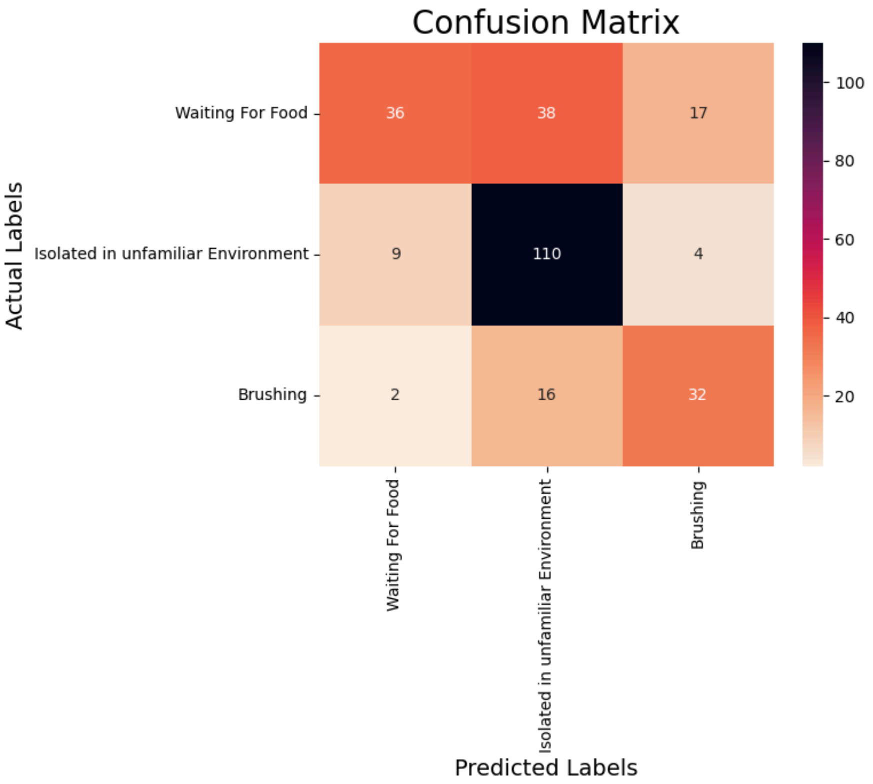
- After training, the model's accuracy is evaluated on the test data.
- Training and testing loss and accuracy are plotted over epochs to visualize the model's performance.



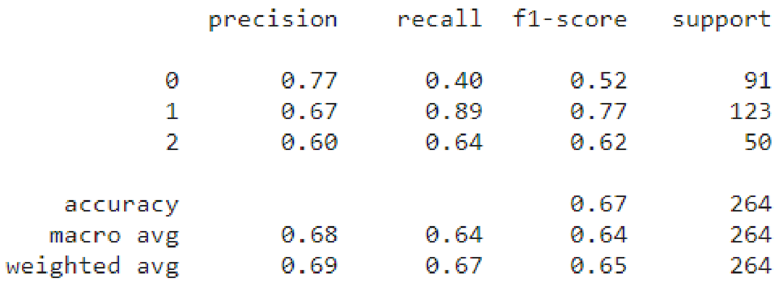
```
Accuracy of our model on test data :  67.42424368858337 %
```

Confusion Matrix and Classification Report:

- Confusion matrix and classification report are generated to evaluate the model's performance on the test data.
- The confusion matrix helps in understanding the model's prediction across different classes.
- The classification report provides precision, recall, and F1-score for each class.



Classification Report

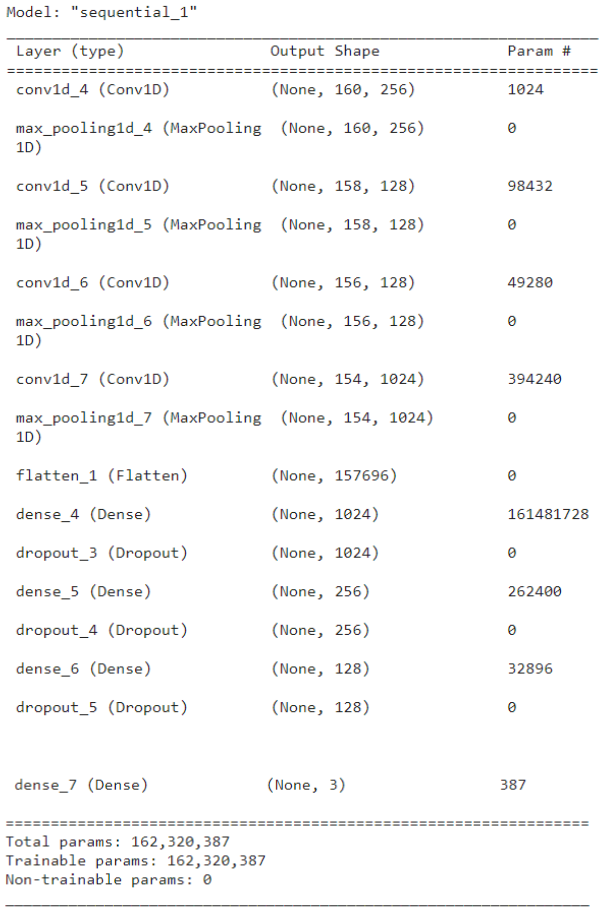


```
              precision    recall  f1-score   support

           0       0.77      0.40      0.52        91
           1       0.67      0.89      0.77       123
           2       0.60      0.64      0.62        50

    accuracy                           0.67       264
   macro avg       0.68      0.64      0.64       264
weighted avg       0.69      0.67      0.65       264
```

Model 2:

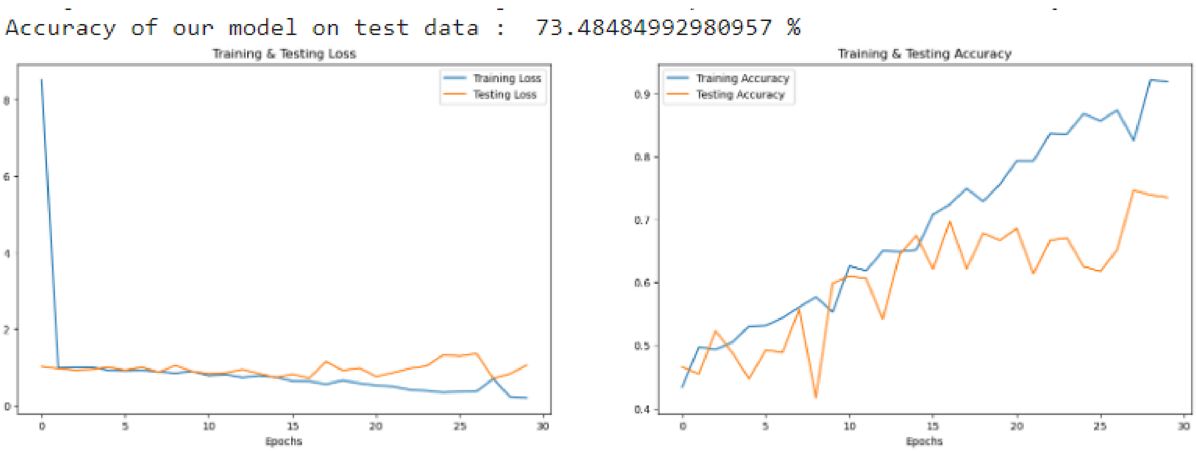
Model Architecture 2:

Similar to Model 1, some parameters are changed



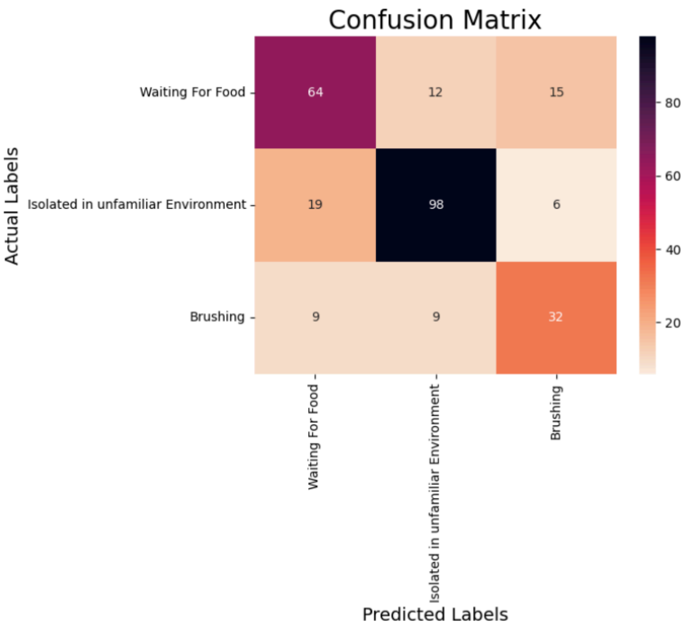
```
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #   
=================================================================
 conv1d_4 (Conv1D)           (None, 160, 256)          1024      
                                                                 
 max_pooling1d_4 (MaxPooling  (None, 160, 256)         0         
 1D)                                                             
                                                                 
 conv1d_5 (Conv1D)           (None, 158, 128)          98432     
                                                                 
 max_pooling1d_5 (MaxPooling  (None, 158, 128)         0         
 1D)                                                             
                                                                 
 conv1d_6 (Conv1D)           (None, 156, 128)          49280     
                                                                 
 max_pooling1d_6 (MaxPooling  (None, 156, 128)         0         
 1D)                                                             
                                                                 
 conv1d_7 (Conv1D)           (None, 154, 1024)         394240    
                                                                 
 max_pooling1d_7 (MaxPooling  (None, 154, 1024)        0         
 1D)                                                             
                                                                 
 flatten_1 (Flatten)         (None, 157696)            0         
                                                                 
 dense_4 (Dense)             (None, 1024)              161481728 
                                                                 
 dropout_3 (Dropout)         (None, 1024)              0         
                                                                 
 dense_5 (Dense)             (None, 256)               262400    
                                                                 
 dropout_4 (Dropout)         (None, 256)               0         
                                                                 
 dense_6 (Dense)             (None, 128)               32896     
                                                                 
 dropout_5 (Dropout)         (None, 128)               0         


 dense_7 (Dense)             (None, 3)                 387       
                                                                 
=================================================================
Total params: 162,320,387
Trainable params: 162,320,387
Non-trainable params: 0
_________________________________________________________________
```

The results of Model 2 are follows:

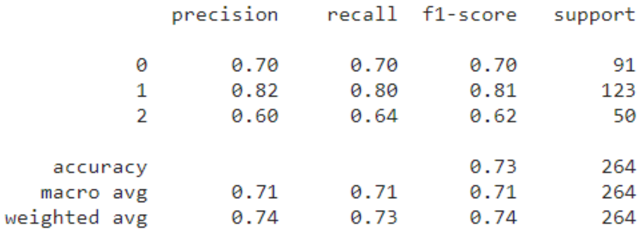


```
Accuracy of our model on test data :  73.48484992980957 %
```

Confusion Matrix



Classification Report



```
              precision    recall  f1-score   support

           0       0.70      0.70      0.70        91
           1       0.82      0.80      0.81       123
           2       0.60      0.64      0.62        50

    accuracy                           0.73       264
   macro avg       0.71      0.71      0.71       264
weighted avg       0.74      0.73      0.74       264
```

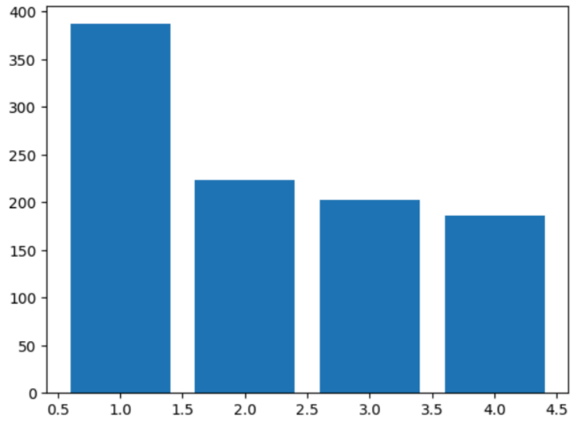
Task 2:

Download any Text Classification dataset (social media text or any short text) and convert it into meaningful image form!!, use Deep CNN models and classify the images.

Implementation:

Data Preprocessing:

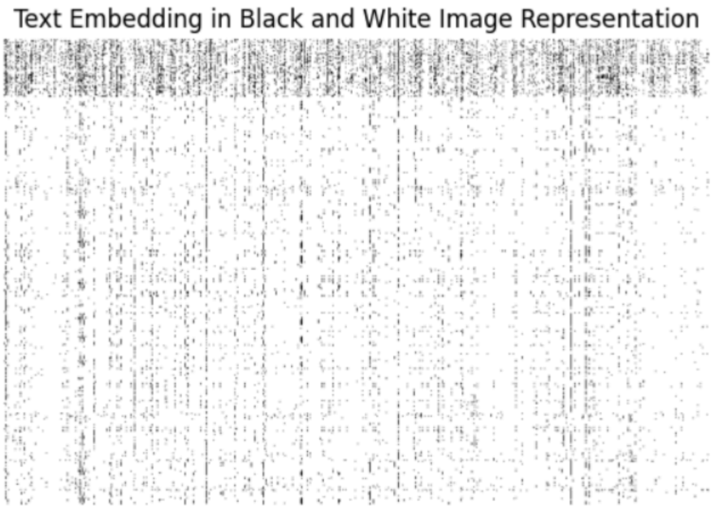
- Calculated the value counts of the target variable



- The dataset contains textual data that needs to be converted into numerical format for machine learning models to process.
- BERT (Bidirectional Encoder Representations from Transformers) is used to convert the text data into word embeddings, which capture the semantic meaning of words in a high-dimensional space.

BERT Integration:

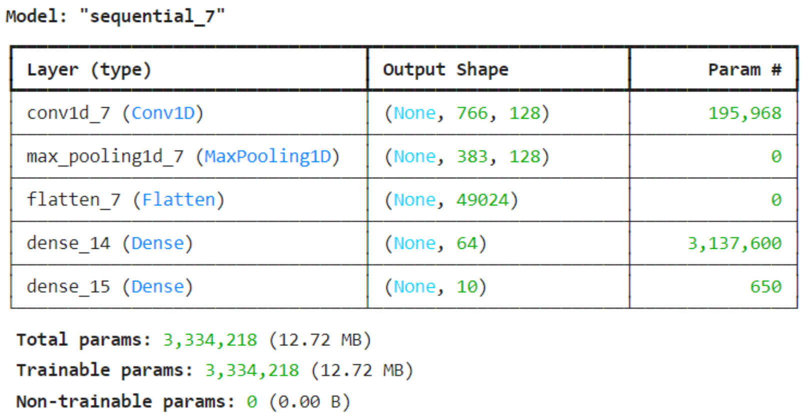
- The BERT model and tokenizer are imported from the Hugging Face Transformers library.
- BERT is used to generate word embeddings for each sentence in the dataset.
- The output from BERT is processed to extract the word embeddings.
- It is visualized with the text embeddings. The embeddings are multiplied with 255 and is converted into a grey scale image



Text Embedding in Black and White Image Representation

Convolutional Neural Network (CNN) Model:

- Two CNN models are implemented to classify the text data.
- Model 1:

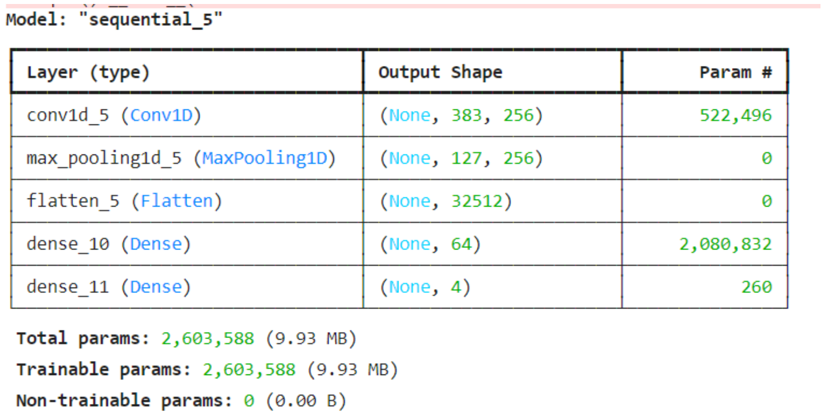


Model: "sequential_7"

Layer (type)	Output Shape	Param #
conv1d_7 (Conv1D)	(None, 766, 128)	195,968
max_pooling1d_7 (MaxPooling1D)	(None, 383, 128)	0
flatten_7 (Flatten)	(None, 49024)	0
dense_14 (Dense)	(None, 64)	3,137,600
dense_15 (Dense)	(None, 10)	650

```
Total params: 3,334,218 (12.72 MB)
Trainable params: 3,334,218 (12.72 MB)
Non-trainable params: 0 (0.00 B)
```

- Model 2:



Model: "sequential_5"

Layer (type)	Output Shape	Param #
conv1d_5 (Conv1D)	(None, 383, 256)	522,496
max_pooling1d_5 (MaxPooling1D)	(None, 127, 256)	0
flatten_5 (Flatten)	(None, 32512)	0
dense_10 (Dense)	(None, 64)	2,080,832
dense_11 (Dense)	(None, 4)	260

```
Total params: 2,603,588 (9.93 MB)
Trainable params: 2,603,588 (9.93 MB)
Non-trainable params: 0 (0.00 B)
```

- The first CNN model (model1) comprises a single Conv1D layer followed by MaxPooling1D, Flatten, and Dense layers.
- The second CNN model (model2) has a similar architecture but with different hyperparameters such as filter size, kernel size, strides, and pool size.

Model Compilation and Training:

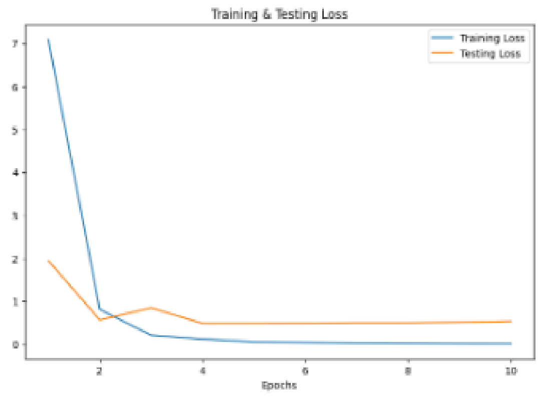
- Both models are compiled using the Adam optimizer and sparse categorical cross-entropy loss function.
- The models are trained using the training data and validated using a portion of the training data.
- The training progress is monitored over multiple epochs.

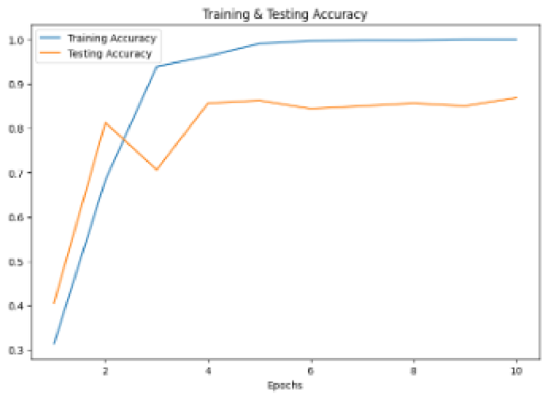
Model Evaluation:

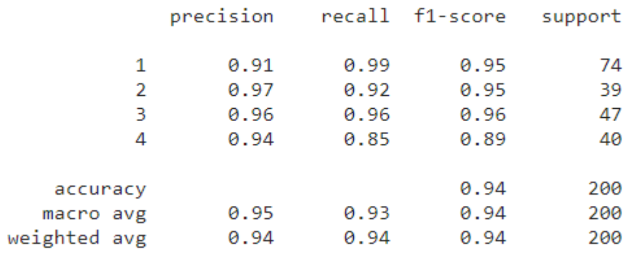
- The trained models are evaluated using the test dataset to measure their performance.
- Evaluation metrics such as accuracy, loss, precision, recall, and F1-score are calculated.
- Confusion matrices are generated to visualize the model's performance in classifying different categories.

Results of Model 1 (128 Filters, 3 kernels, 1 Stride, Max Pool Size = 2):

- Test accuracy: 0.9400

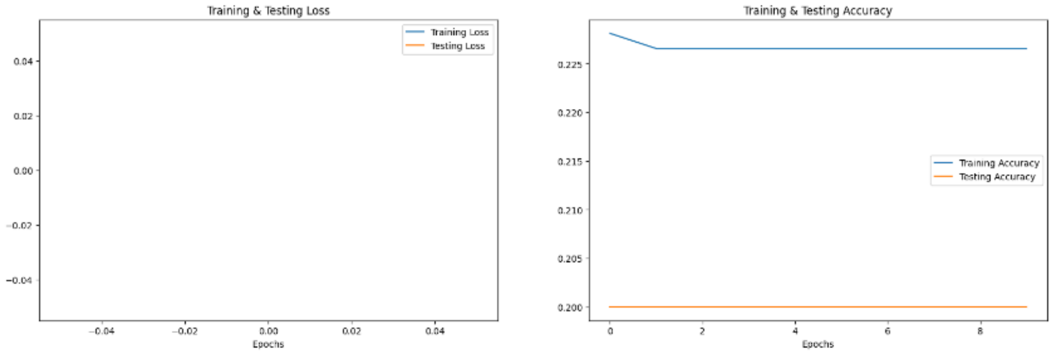


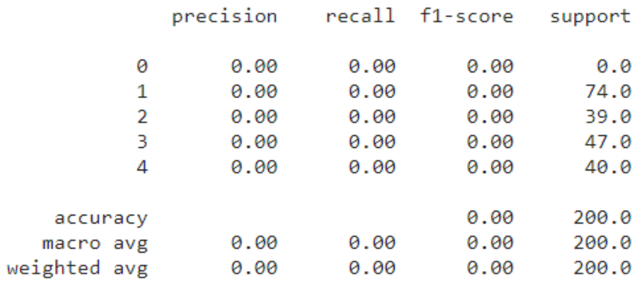




```
              precision    recall  f1-score   support

           1       0.91      0.99      0.95        74
           2       0.97      0.92      0.95        39
           3       0.96      0.96      0.96        47
           4       0.94      0.85      0.89        40

    accuracy                           0.94       200
   macro avg       0.95      0.93      0.94       200
weighted avg       0.94      0.94      0.94       200
```

Results of Model 2





```
              precision    recall  f1-score   support

           0       0.00      0.00      0.00       0.0
           1       0.00      0.00      0.00      74.0
           2       0.00      0.00      0.00      39.0
           3       0.00      0.00      0.00      47.0
           4       0.00      0.00      0.00      40.0

    accuracy                           0.00     200.0
   macro avg       0.00      0.00      0.00     200.0
weighted avg       0.00      0.00      0.00     200.0
```

Conclusion:

Model 1 performed well as compared to Model 2. As the parameters change, the metrics also varies