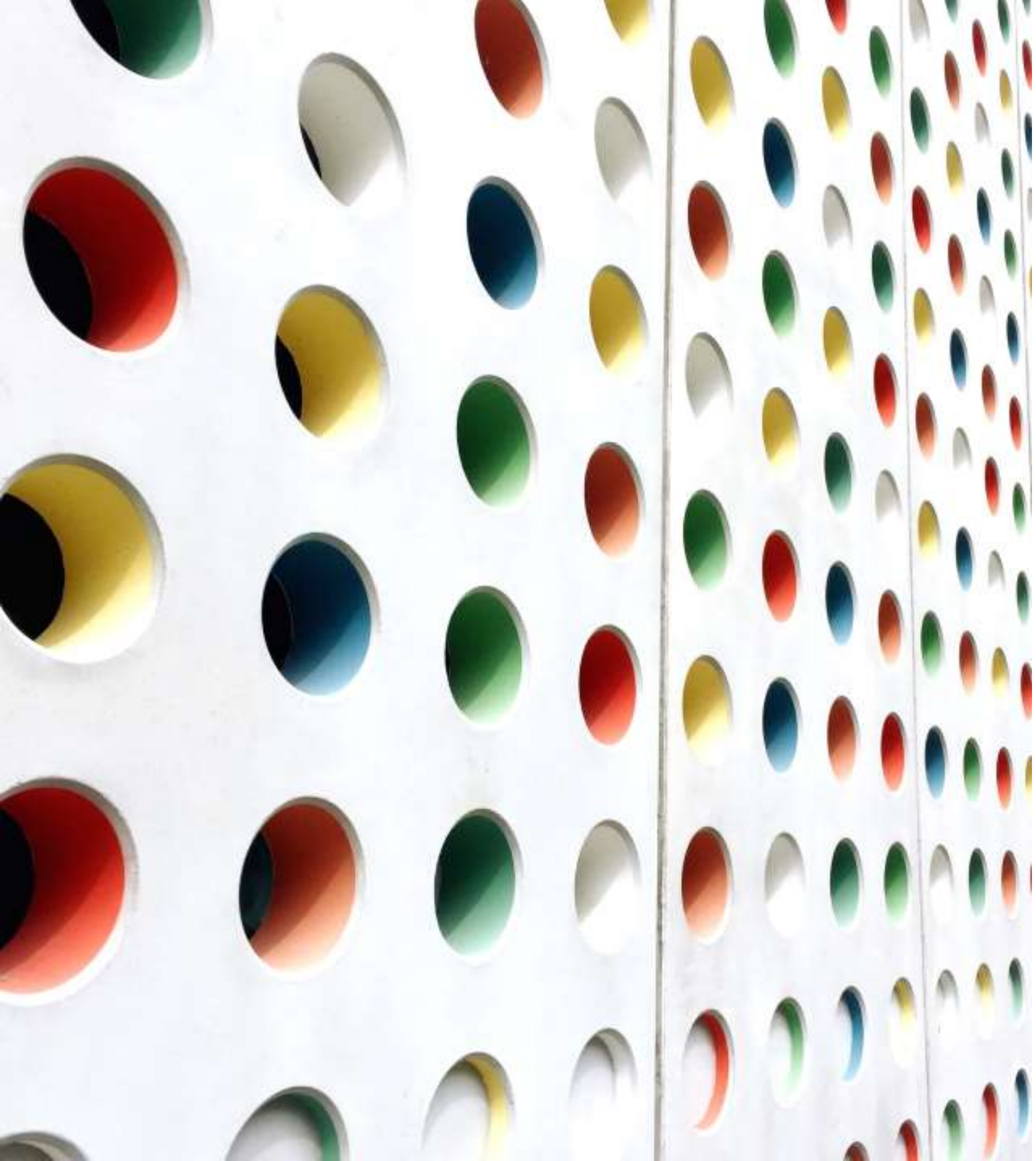


EMOTION DETECTION

CHARAN SUMANTH PULLETI

CONTENTS

INTRODUCTION

- Emotion detection, also known as sentiment analysis, is the process of identifying and categorizing human emotions expressed in text or speech data. In this project, we aim to develop machine learning models capable of accurately detecting and classifying emotions in text data.
- I employed three different models: LSTM (Long Short-Term Memory), BERT (Bidirectional Encoder Representations from Transformers), and CNN (Convolutional Neural Network), to explore various approaches to emotion detection.

PROJECT OBJECTIVE

- The primary objective of this project is to develop a model capable of accurately detecting and classifying emotions expressed in text data.
- By achieving high accuracy in emotion detection tasks, we seek to enhance understanding and analysis of human emotions as expressed through written communication.
- My aim was to create a system that can classify text into a set of predefined emotion categories, such as joy, sadness, anger, neutral etc., with high accuracy.
- This system holds significant potential for applications across various domains, including customer feedback analysis, healthcare, market research, and sentiment analysis, offering valuable insights into the emotional content of textual information.

DATASET OVERVIEW

- The dataset utilized in this project consists of textual data annotated with corresponding emotion labels. It comprises a total of 16,000 samples distributed across six emotion classes: sadness, joy, love, anger, fear, and surprise.
- My dataset contains three text files training, test and validation text files. Prior to model training, several preprocessing steps were applied to the dataset. These steps include removing duplicate entries to ensure data integrity and consistency.

Train Data

```
print("Training Data:")
print(train_data.head())
```

```
Training Data:
                                                text  label
0                            i didnt feel humiliated      0
1  i can go from feeling so hopeless to so damned...      0
2   im grabbing a minute to post i feel greedy wrong      3
3  i am ever feeling nostalgic about the fireplac...      2
4                               i am feeling grouchy      3
```

Test Data

```
print(test_data.head())
```

```
                                                text  label
0  im feeling rather rotten so im not very ambiti...      0
1          im updating my blog because i feel shitty      0
2  i never make her separate from me because i do...      0
3  i left with my bouquet of red and yellow tulip...      1
4    i was feeling a little vain when i did this one      0
```

Validation Data

```
print(val_data.head())
```

```
                                                text  label
0  im feeling quite sad and sorry for myself but ...      0
1  i feel like i am still looking at a blank canv...      0
2                     i feel like a faithful servant      2
3                  i am just feeling cranky and blue      3
4     i can have for a treat or if i am feeling festive   1
```

```
labels_dict = {
    0: 'sadness',
    1: 'joy',
    2: 'love',
    3: 'anger',
    4: 'fear',
    5: 'surprise'
}

train_data['description'] = train_data['label'].map(labels_dict)

print("Updated training data with description:")
print(train_data.head())
```

```
Updated training data with description:
                                                text  label description
0                            i didnt feel humiliated      0     sadness
1  i can go from feeling so hopeless to so damned...      0     sadness
2   im grabbing a minute to post i feel greedy wrong      3       anger
3  i am ever feeling nostalgic about the fireplac...      2        love
4                               i am feeling grouchy      3       anger
```

DATA PREPROCESSING

- Additionally, text data underwent tokenization and padding processes to convert text inputs into numerical sequences suitable for training machine learning models.
- Tokenization involves breaking down text into individual words or sub words, while padding ensures that all sequences have uniform length, facilitating efficient model training.
- These preprocessing steps are essential for preparing the dataset for input into machine learning models, ensuring optimal performance and accuracy during training and evaluation phases.

MODEL ARCHITECTURES

◦ In this project, three distinct models were employed for emotion detection:

1. LSTM (Long Short-Term Memory)
2. BERT (Bidirectional Encoder Representations from Transformers)
3. CNN (Convolutional Neural Network).

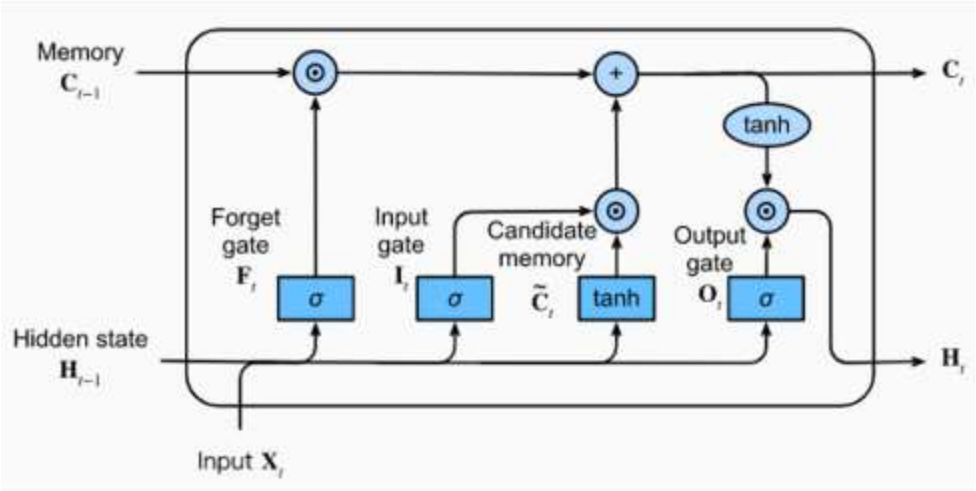
◦ Each model offers unique advantages and architectures suited for processing textual data and extracting meaningful features.

LSTM MODEL

The LSTM model is a type of recurrent neural network (RNN) designed to handle sequence data, making it particularly well-suited for processing text.

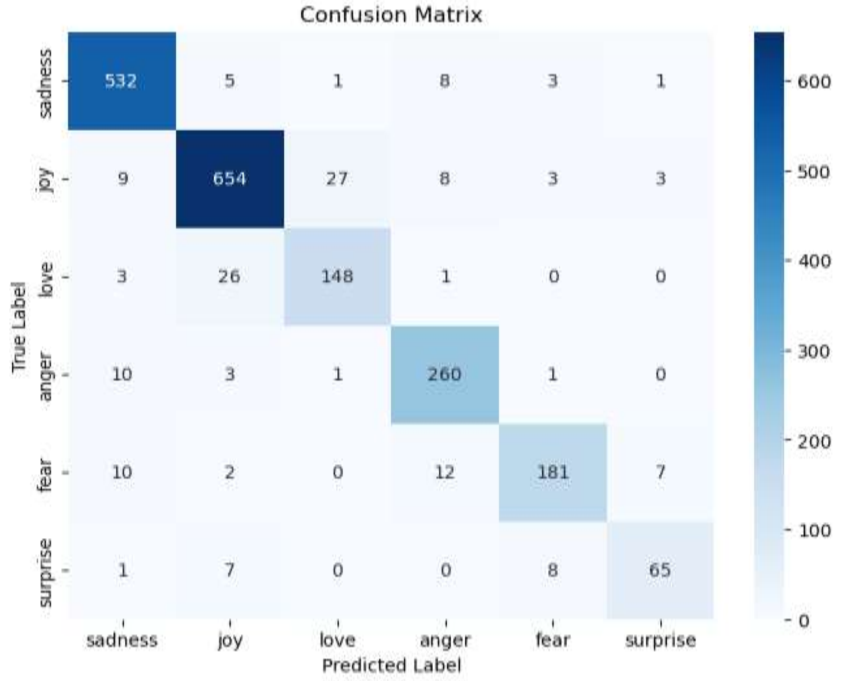
It consists of multiple LSTM cells that can retain information over long sequences, enabling the model to capture dependencies and patterns in the input data.

The architecture includes an embedding layer for converting words into dense numerical vectors, followed by LSTM layers to process sequential data.



1. LSTM Model

In [18]:
```
from keras.models import Sequential
from keras.layers import LSTM, Dense, Embedding, SpatialDropout1D
from keras.optimizers import Adam

num_words = len(tokenizer.word_index) + 1
max_sequence_length = max(len(seq) for seq in X_train_pad)

def create_lstm_model():
    model = Sequential()
    model.add(Embedding(input_dim=num_words, output_dim=100, input_length=max_sequence_length))
    model.add(SpatialDropout1D(0.2))
    model.add(LSTM(64, dropout=0.2, recurrent_dropout=0.2))
    model.add(Dense(6, activation='softmax'))

    optimizer = Adam(learning_rate=0.001)
    model.compile(optimizer=optimizer, loss='sparse_categorical_crossentropy', metrics=['accuracy'])
    return model

lstm_model = create_lstm_model()

history = lstm_model.fit(X_train_pad, y_train, epochs=10, batch_size=64, validation_data=(X_val_pad, y_val))

test_loss, test_accuracy = lstm_model.evaluate(X_test_pad, y_test)
print("Test Loss:", test_loss)
print("Test Accuracy:", test_accuracy)
```




```
Epoch 9/10
250/250 [==============================] - 63s 251ms/step - loss: 0.0482 - accuracy: 0.9834 - val_loss: 0.2811 - val_accuracy:
0.9190
Epoch 10/10
250/250 [==============================] - 63s 251ms/step - loss: 0.0440 - accuracy: 0.9859 - val_loss: 0.2789 - val_accuracy:
0.9200
63/63 [==============================] - 2s 27ms/step - loss: 0.2773 - accuracy: 0.9110
Test Loss: 0.27731403708457947
Test Accuracy: 0.9110000133514404
```

```
63/63 [==============================] - 2s 24ms/step
              precision    recall  f1-score   support

           0       0.94      0.97      0.95       550
           1       0.94      0.93      0.93       704
           2       0.84      0.83      0.83       178
           3       0.90      0.95      0.92       275
           4       0.92      0.85      0.89       212
           5       0.86      0.80      0.83        81

    accuracy                           0.92      2000
   macro avg       0.90      0.89      0.89      2000
weighted avg       0.92      0.92      0.92      2000
```

BERT MODEL

- BERT is a transformer-based model renowned for its effectiveness in natural language processing tasks.
- Unlike traditional models that process text in a unidirectional manner, BERT leverages bidirectional attention mechanisms to capture context from both preceding and succeeding words in a sentence.
- The architecture consists of multiple transformer layers, each comprising self-attention and feed-forward neural network sublayers.
- BERT can effectively learn contextual representations of words in a sentence, making it highly suitable for tasks such as sentiment analysis and emotion detection.

BERT MODEL OVERVIEW

- Preprocessing: Utilized the BERT tokenizer from the transformers library to tokenize text data.
- Input Data Preparation: Processed the training, validation, and test datasets using the BERT tokenizer, ensuring padding, truncation, and a maximum sequence length of 100 tokens.
- Data Dimensions:
- Train inputs shape: torch.Size([15999, 87])
- Validation inputs shape: torch.Size([2000, 69])
- Test inputs shape: torch.Size([2000, 66])

BERT Model

```
from transformers import BertTokenizer

tokenizer = BertTokenizer.from_pretrained('bert-base-uncased')

train_inputs = tokenizer(list(train_data['text']), padding=True, truncation=True, max_length=100, return_tensors="pt")
val_inputs = tokenizer(list(val_data['text']), padding=True, truncation=True, max_length=100, return_tensors="pt")
test_inputs = tokenizer(list(test_data['text']), padding=True, truncation=True, max_length=100, return_tensors="pt")

print("Train inputs shape:", train_inputs['input_ids'].shape)
print("Validation inputs shape:", val_inputs['input_ids'].shape)
print("Test inputs shape:", test_inputs['input_ids'].shape)
```

```
Train inputs shape: torch.Size([15999, 87])
Validation inputs shape: torch.Size([2000, 69])
Test inputs shape: torch.Size([2000, 66])
```

MODEL ARCHITECTURE AND TRAINING

- Model Initialization: Used BertForSequenceClassification from the transformers library, initialized with the 'bert-base-uncased' pre-trained model.
- Optimizer: Employed the AdamW optimizer with a learning rate of 1e-5.
- Loss Function: Utilized Cross-Entropy Loss for classification tasks.
- Training Process: Trained the BERT model for one epoch using a batch size of 32, with input sequences shuffled during training.
- Validation: Evaluated model performance on a separate validation dataset, achieving an accuracy of 92%.

Defining Loss Function

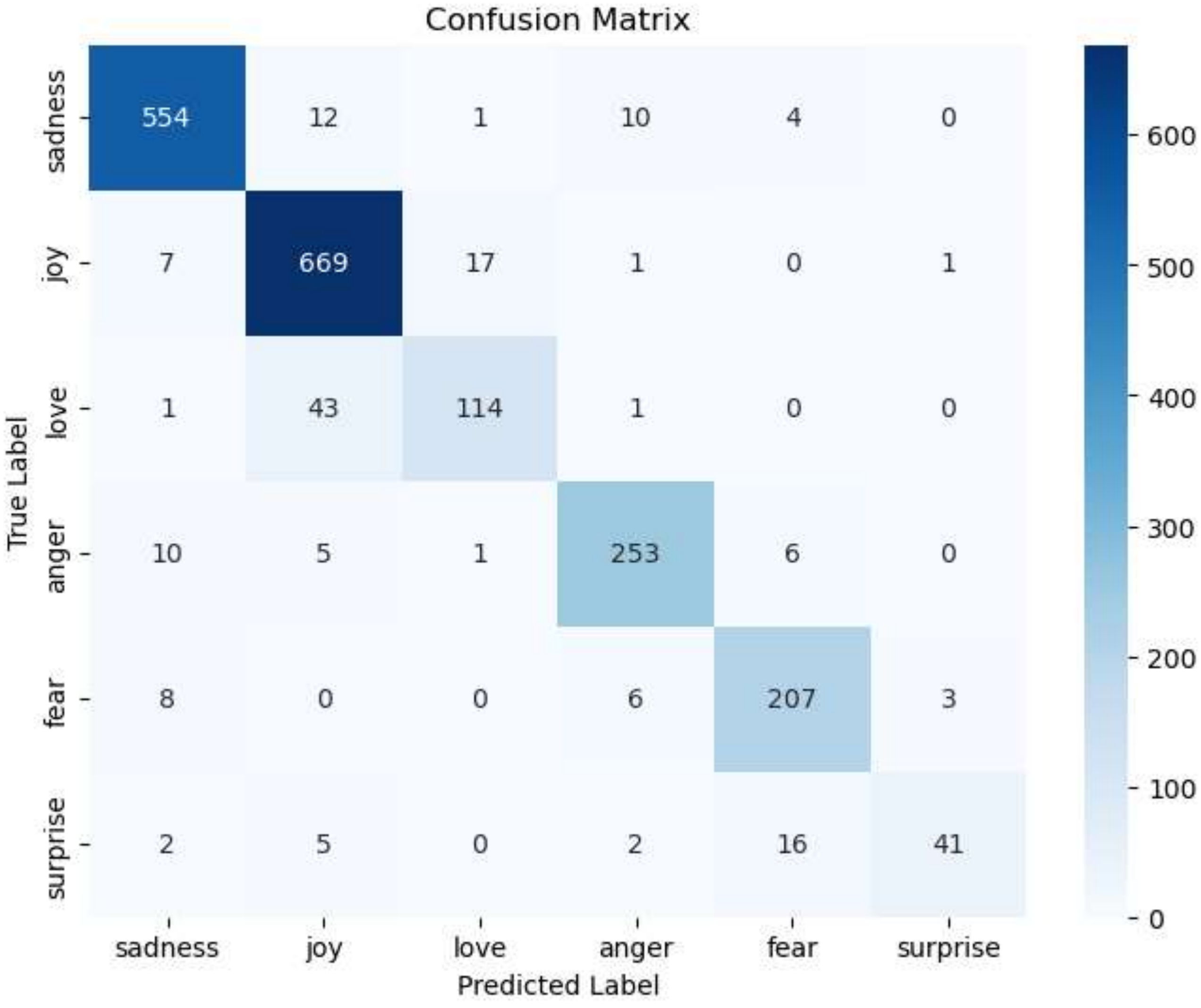
```python
import torch
import torch.nn as nn
from transformers import BertForSequenceClassification, AdamW

model = BertForSequenceClassification.from_pretrained('bert-base-uncased', num_labels=num_classes)

optimizer = AdamW(model.parameters(), lr=1e-5, eps=1e-8)


loss_fn = nn.CrossEntropyLoss()
```

```python
from sklearn.metrics import accuracy_score

predicted_labels_bert = val_predicted_labels

accuracy_bert = accuracy_score(y_val, predicted_labels_bert)

print("BERT Model Accuracy:", accuracy_bert)
```

```
BERT Model Accuracy: 0.92
```

```
print("Training complete.")
```

```
Some weights of BertForSequenceClassification were not initialized from the model checkpoint at bert-base-uncased and are newly
initialized: ['classifier.weight', 'classifier.bias']
You should probably TRAIN this model on a down-stream task to be able to use it for predictions and inference.
Epoch 1/1
 20%|██████████                                        | 101/500 [48:02<3:10:12, 28.60s/it]
  Batch 100/500 - Avg. Loss: 1.5175
 40%|████████████████████                              | 201/500 [1:35:55<2:26:05, 29.31s/it]
  Batch 200/500 - Avg. Loss: 1.2621
 60%|██████████████████████████████                    | 301/500 [1:57:04<36:08, 10.90s/it]
  Batch 300/500 - Avg. Loss: 1.0502
 80%|████████████████████████████████████████          | 401/500 [2:15:51<23:02, 13.96s/it]
  Batch 400/500 - Avg. Loss: 0.8889
100%|██████████████████████████████████████████████████| 500/500 [2:34:52<00:00, 18.59s/it]
  Average training loss: 0.7710
  Validation Loss: 0.2559
Training complete.
```



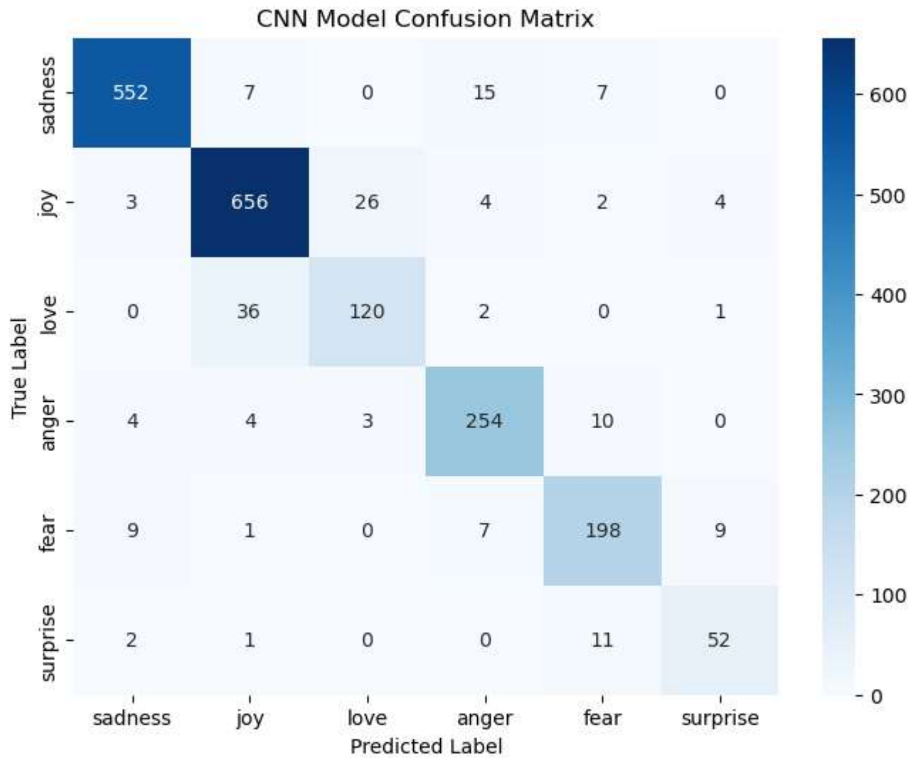
Classification Report:

	precision	recall	f1-score	support
sadness	0.95	0.95	0.95	581
joy	0.91	0.96	0.94	695
love	0.86	0.72	0.78	159
anger	0.93	0.92	0.92	275
fear	0.89	0.92	0.91	224
surprise	0.91	0.62	0.74	66
accuracy			0.92	2000
macro avg	0.91	0.85	0.87	2000
weighted avg	0.92	0.92	0.92	2000

CNN MODEL

- The CNN model for emotion classification leverages cutting-edge deep learning techniques to accurately discern emotional states from textual data.
- Through a combination of convolutional filters and pooling layers, the model captured meaningful patterns and features within the input text, enabling it to make informed predictions about the underlying emotions expressed.
- This architecture allows the model to process textual data efficiently, making it well-suited for real-world applications where rapid and accurate emotion detection is paramount.

```python
from keras.models import Sequential
from keras.layers import Embedding, Conv1D, GlobalMaxPooling1D, Dense, Dropout

def create_cnn_model():
    model = Sequential()
    model.add(Embedding(input_dim=max_vocab_size, output_dim=100, input_length=max_seq_length))
    model.add(Conv1D(filters=64, kernel_size=3, padding='same', activation='relu'))
    model.add(GlobalMaxPooling1D())
    model.add(Dense(128, activation='relu'))
    model.add(Dropout(0.5))
    model.add(Dense(num_classes, activation='softmax'))

    model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])
    return model

cnn_model = create_cnn_model()
```

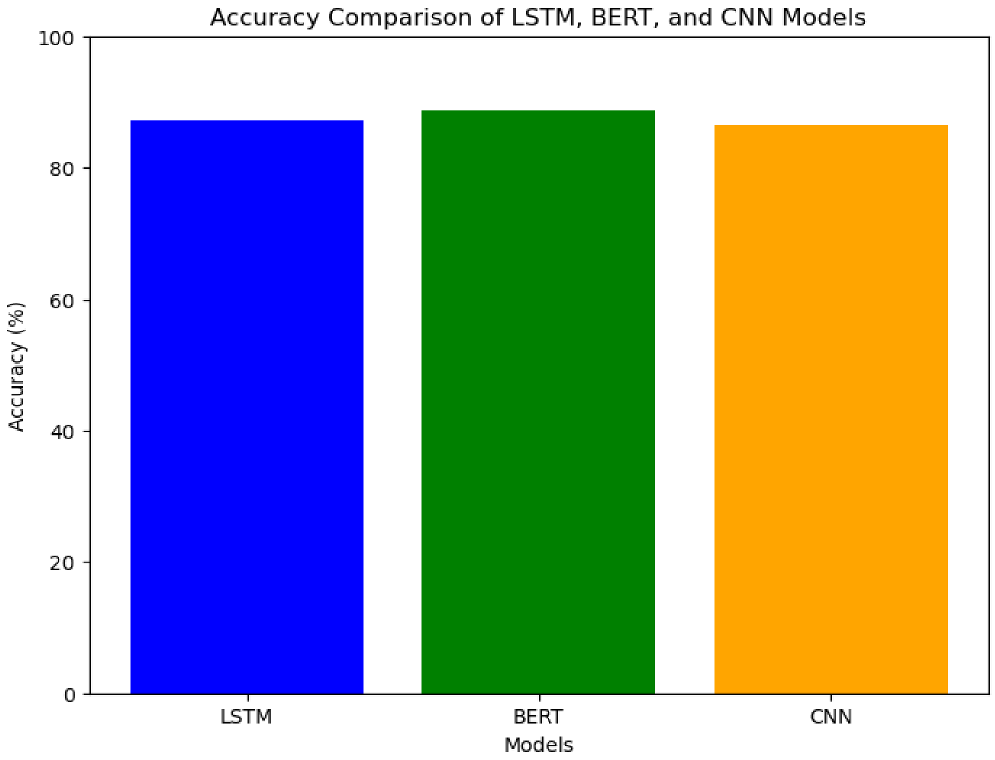


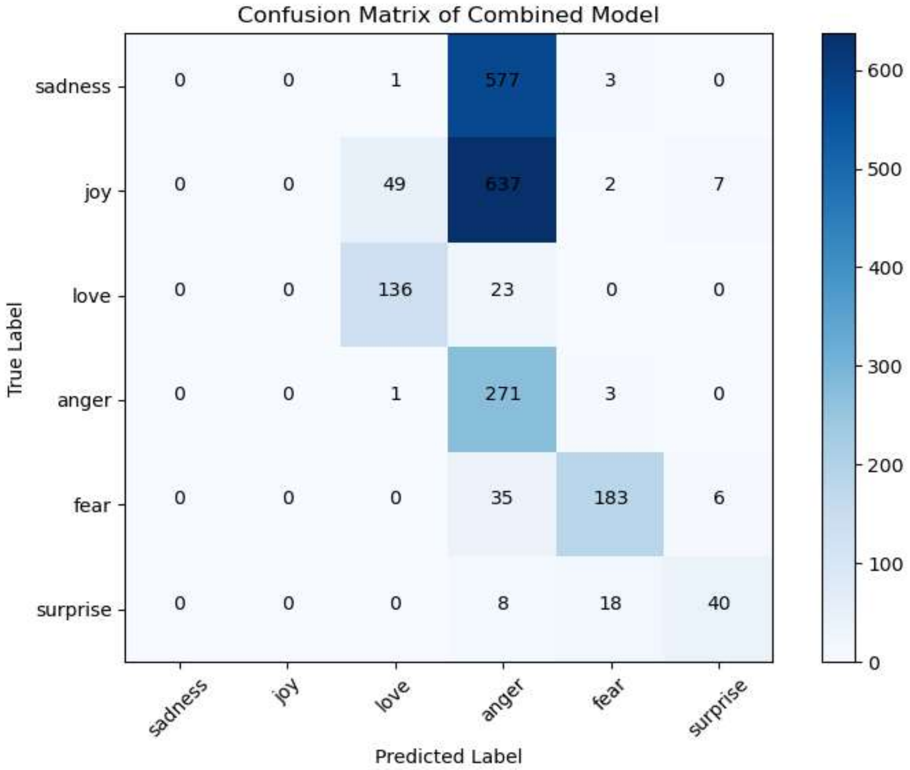

```
63/63 [==============================] - 0s 4ms/step
CNN Model Classification Report:
```

	precision	recall	f1-score	support
sadness	0.97	0.95	0.96	581
joy	0.93	0.94	0.94	695
love	0.81	0.75	0.78	159
anger	0.90	0.92	0.91	275
fear	0.87	0.88	0.88	224
surprise	0.79	0.79	0.79	66
accuracy			0.92	2000
macro avg	0.88	0.87	0.88	2000
weighted avg	0.92	0.92	0.92	2000

```
Epoch 9/10
250/250 [==============================] - 3s 13ms/step - loss: 0.0244 - accuracy: 0.9922 - val_loss: 0.2716 - val_accuracy: 0.
9225
Epoch 10/10
250/250 [==============================] - 3s 13ms/step - loss: 0.0223 - accuracy: 0.9923 - val_loss: 0.2890 - val_accuracy: 0.
9205
63/63 [==============================] - 0s 4ms/step - loss: 0.2866 - accuracy: 0.9160
Test Loss: 0.2865625023841858
Test Accuracy: 0.9160000085830688
```

MODEL COMPARISON





ADVANTAGES

- Customer Service: Real-time emotion detection can enhance customer service by enabling immediate response to customer sentiments.
- Educational Technology: Emotion detection can personalize learning experiences by adapting content based on students emotional states in real-time.
- Mental Health Monitoring: Real-time emotion detection can aid in monitoring individuals mental health by providing timely interventions and support.
- Market Research: Emotion detection in real-time can provide valuable insights into consumer preferences and reactions to products or advertisements.
- Human-Computer Interaction: Real-time emotion detection can improve user experience by enabling devices to adapt their responses based on users emotional cues.

CONCLUSION

- In conclusion, the project successfully developed models for accurately detecting and classifying emotions expressed in text data. Through the implementation of LSTM, BERT, and CNN architectures, each model showcased robust performance in emotion classification.
- The project's LSTM model boasts a commendable 91% accuracy, particularly excelling in predicting joy and sadness, though it faces challenges with fear and surprise. Similarly, BERT demonstrates strong performance with 92% accuracy, showcasing robustness in recognizing various emotions, while the CNN model achieves an impressive overall accuracy of 91%, particularly excelling in classifying sadness and joy.
- BERT's greater accuracy is probably a result of its skill at efficiently capturing contextual information using language representations currently provided. Compared to LSTM and CNN, this enables it to recognize complexities in text input more effectively, resulting in more accurate emotion classification.

REFERENCES

- Deep Learning Approaches: The use of deep learning techniques has increased dramatically in response to recent developments in emotion recognition. CNNs (LeCun etal., 1998) and LSTM (Hochreiter & Schmidhuber, 1997) are two examples of models that have shown remarkable ability in extracting contextual and hierarchical features from text data.
- Zhao et al. (2020) "BERT for Emotion Recognition and Classification in Twitter Sentiment Analysis": The study looks into how well BERT-based models work in Twitter sentiment analysis to identify emotions.
- Transformer-based Architectures: Natural language processing tasks, such as sentiment analysis and emotion identification, have been revolutionized by transformer-based models like BERT (Devlin et al., 2018) and RoBERTa (Liu et al., 2019).

THANK YOU