Project

Emotion Detection Based on Image and Text

Report submitted
by

Mishika Sardana 102317282
Ragini Sharma 102317266

Submitted to
Ms. Kanika

THAPAR INSTITUTE OF ENGINEERING AND TECHNOLOGY
(A DEEMED TO BE UNIVERSITY), PATIALA, PUNJAB
INDIA

INDEX

ABSTRACT

Emotion is a key component of human communication, affecting decision-making, behavior, and interaction. As Artificial Intelligence continues to evolve, the development of systems that are capable of detecting and interpreting emotions has become ever more crucial, particularly for uses such as mental health monitoring, customer service, and human-computer interaction. Most current models, however, are hampered by their use of a single modality—text or facial expression—and as such are insufficient for properly interpreting the complete range of human emotion.

This project proposes a multi-modal emotion detection system that utilizes both text and image-based data to achieve a more comprehensive and precise emotion classification. Employing grayscale face expression images from the FER2013 dataset and emotion-labeled tweets from the emotion_sentiment_dataset.csv dataset, the system uses a mixture of supervised and unsupervised machine learning algorithms such as Support Vector Machine (SVM), Logistic Regression, Random Forest, Naïve Bayes, and TF-IDF, XGBoost.

The model classifies emotions into categories such as happy, sad, angry, surprise, fear, and neutral, allowing for subtle detection through data fusion. It solves significant problems of conventional systems like low accuracy in uncontrolled environments, poor generalization, and absence of real-time performance. This project also focuses on responsible deployment of AI, taking into account privacy, data security, and ethical concerns while developing and testing the system. The outcome is a context-sensitive, and multi-sensor emotion detection model with broad scope applications.

INTRODUCTION

Understanding human emotion is imperative for the development of machines that can interact with users effectively. Emotions are complex and multidimensional, often expressed through a combination of facial expressions, voice, body language, and textual content. However, most existing emotion detection systems rely on a single modality, such as facial expressions, or text, which limits their ability to capture the full spectrum of human emotions. These systems often fail to account for the interplay between different modalities, leading to reduced accuracy and reliability. Additionally, they face challenges such as low accuracy in diverse or uncontrolled environments, inability to operate in real time, and a lack of effective multi-sensor data fusion. These limitations hinder their practical application in real-world scenarios, such as mental health monitoring, customer feedback analysis, and human-computer interaction.

The problem is further compounded by the need to analyze context-rich data, such as images, videos, and texts, which often contain subtle cues essential for accurate emotion detection. For instance, a person's facial expression in a video might convey one emotion, while their choice of words might suggest another. Failing to consider these nuances can lead to misinterpretations. Moreover, deploying emotion detection systems on mobile devices or edge computing platforms introduces challenges related to computational limitations and energy efficiency, which must be addressed to enable real-time processing. Ethical concerns, such as privacy and data security, also arise when handling sensitive user information, making it crucial to build trust and ensure responsible deployment.

This project aims to address these challenges by proposing a system capable of real-time emotion recognition using multiple modalities, such as text, and visual data. By integrating these modalities, the system will provide a more comprehensive and accurate understanding of human emotions while prioritizing user privacy and ethical considerations. These advancements will improve mental health detection, customer response evaluation, and human-computer interaction systems, paving the way for more reliable and context-aware emotion detection tech

PROBLEM STATEMENT

Emotion perception by machines is a persistent difficult task because emotions are subjective and multidimensional, and typically conveyed through delicate combinations of text, facial expressions, tone, and body posture. Most systems that currently exist are single-modality input—image or text—and thus not adequate to understand the whole emotional context, particularly in real-world environments that tend to be typically noisy, ambiguous, or heterogeneous. This context-unawareness results in lower model accuracy and reduces the system's ability to generalize to a broad set of user populations.

Text-based models are liable to fail in detecting sarcasm, slang, or cultural inferences, whereas image-based models can struggle with low-resolution inputs, occluded faces, or poor lighting. Additionally, text inputs are vague and lack emotional nuance in the absence of vocal tone and facial expression. Multilingual inputs and informal language also challenge textual emotion detection. Similarly, facial emotion recognition models can struggle with dynamic expressions or micro-expressions that flash fleeting and subtly.

In addition, the absence multi-modal integration restricts their applications to static areas such as sentiment analysis, opinion mining, and text classification. For the majority of these domains, precise emotion detection in real-time is crucial for emotional and effective communication. Furthermore, the absence of synchronizing across different modalities leads to disconnected insights and impedes more integrated emotional evaluation.

Besides, deployment of such systems on edge or mobile devices is a resource and latency issue, while deployment of personal data like facial images and tweets is a privacy and ethical issue. Data storage, user consent, and algorithmic transparency are some of the concerns that have to be dealt with to ensure trust and compliance with ethical AI practices. Therefore, it is necessary to develop a strong, real-time, multi-modal emotion recognition system with both visual and textual modalities fusion to improve the detection accuracy as well as ensure safe use of AI.

OBJECTIVE

The main objective of this project is to develop an AI-powered emotion detection model that integrates facial expression recognition and text sentiment analysis to provide a comprehensive understanding of human emotions. The specific objectives are:

Integration of multiple modes:
Combine visual and textual inputs to create a unified emotion detection system capable of processing multiple input types concurrently.
Improve the accuracy and robustness of emotion detection by integrating knowledge from both modalities, resulting in better performance in real-world situations.
Incorporate context-aware emotion interpretation, considering both facial expressions and language tone, to accurately identify and interpret subtle or conflicting emotional signals.

Data Application:
Employ the FER dataset to categorize facial expressions into 7 distinct emotions: anger, disgust, fear, happiness, sadness, surprise, and neutrality.
Employ the emotion_sentiment_dataset.csv dataset to identify emotions in texts, initially encompassing up to 13 emotion labels (e.g., happy, sad, annoyed, excited), with the possibility of class balancing or reduction to optimize model performance.
Before training the model, perform data preprocessing steps such as normalization, tokenization (for text), resizing (for images), and addressing imbalanced data classes to enhance the model's performance.

Code execution and evaluation:
Utilize and assess multiple machine learning techniques:
Support vector machine (svm) is a popular choice due to its ability to handle high-dimensional and sparse datasets, making it particularly effective for text classification tasks.

Logistic regression is a statistical method used to make predictions in both binary and multiclass scenarios, allowing for comparisons between different baseline predictions.
Random forest, a technique used for ensemble-based classification, enhances accuracy and minimizes variance.
Naïve Bayes is a fast and scalable text classification method, particularly effective for handling large amounts of textual data.
TF-IDF transforms raw text into numerical feature vectors.
XGBoost uses structured feature sets like TF-IDF vectors.

Assessment of Our Model :
Evaluate and assess models by utilizing key performance metrics such as accuracy, precision, recall, f1-score, and confusion matrices.
Evaluate the performance of different models and their combined results in various scenarios.

Issues of morality and confidentiality:
To avoid unauthorized access to personal information, such as facial photos and individual tweets, implement anonymization, encryption, and user consent protocols.
Address concerns about bias, fairness, and transparency, ensuring that the system functions consistently and without discrimination across various demographic groups.

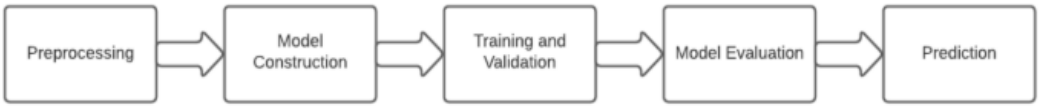
Real-world uses:
Develop the system to be utilized in:
Keeping track of mental health and emotions in therapy apps or well-being dashboards.
Evaluation of customer feedback and social media sentiment to improve brand and service performance.
Intelligent human-computer interaction, like emotionally responsive virtual assistants, chatbots, or e-learning systems that exhibit empathy.
Potential integration in automated surveillance, gaming, or interactive storytelling, enhancing realism and personalization.

METHODOLOGY

Planning:

- Basic steps in constructing a Machine Learning model:
  - ➢ Data Collection:- The quantity & quality of your data dictate how accurate our model is.
  - ➢ Data Preparation:- Wrangling data and preparing it for training.
  - ➢ Choose a Model:- Algorithm like SVM is used.
  - ➢ Train the Model:- The goal of training is to answer a question or make a prediction correctly as often as possible.

- Workflow:



- Import the libraries:
  - ➢ Libraries like Numpy, Pandas, matplotlib.pyplot, seaborn, sklearn.neighbors, etc.

- Dataset:
  We will use emotion detection datasets from reliable sources such as:

  Fer dataset:
  https://www.kaggle.com/datasets/ashishpatel26/facial-expression-recognitionferchallenge

  About Dataset
  - ➢ The data consists of 48x48 pixel grayscale images of faces. The faces have been automatically registered so that the face is more or less centred and occupies about the same amount of space in each image.
  - ➢ The task is to categorize each face based on the emotion shown in the facial expression into one of seven categories (0=Angry,

1=Disgust, 2=Fear, 3=Happy, 4=Sad, 5=Surprise, 6=Neutral). The dataset consists of 35887 examples of images.

Emotion detection from text:
https://www.kaggle.com/datasets/simaanjali/emotion-analysis-based-on-text

About Dataset:

- ➢ The data is basically a collection of text annotated with the emotions behind them. We have three columns: one is unnamed, text, and emotion. In "text" we have the raw text. In "emotion" we have the emotion behind the text.
- ➢ This public domain dataset is collected from data.world platform.
- ➢ The data that we have is having 13 different emotion 839555 records. So it's challenging to build an efficient multiclass classification model. We may need to logically reduce the number of classes here and use some advanced methods to build an efficient model.

Code Explanation:

- Importing Libraries and ML Tools

```
import math
import numpy as np
import pandas as pd

import scikitplot
import seaborn as sns
from matplotlib import pyplot

from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder
from sklearn.metrics import classification_report

import tensorflow as tf
from tensorflow.keras import optimizers
from tensorflow.keras.datasets import mnist
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Flatten, Dense, Conv2D, MaxPooling2D
from tensorflow.keras.layers import Dropout, BatchNormalization, LeakyReLU, Activation
from tensorflow.keras.callbacks import Callback, EarlyStopping, ReduceLROnPlateau
from tensorflow.keras.preprocessing.image import ImageDataGenerator

from keras.utils import np_utils
```

```
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import classification_report, confusion_matrix
from sklearn.preprocessing import StandardScaler
```

```
from sklearn.pipeline import Pipeline
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.linear_model import LogisticRegression
```

```
from sklearn.ensemble import RandomForestClassifier
from sklearn.pipeline import Pipeline
from sklearn.feature_extraction.text import TfidfVectorizer
```

```
from sklearn.svm import SVC
```

```
from xgboost import XGBClassifier
```

```
from sklearn.naive_bayes import MultinomialNB
from sklearn.pipeline import Pipeline
from sklearn.feature_extraction.text import TfidfVectorizer
```

- NumPy is used for working with Numerical values as it makes it easy to apply mathematical functions.
- Pandas is mostly used for data analysis tasks in Python.
- Matplotlib is a python library used to create 2D graphs and plots by using python scripts.

❖ Seaborn is a Python data visualization library based on matplotlib. It provides a high- level interface for drawing attractive and informative statistical graphics.

- Importing Datasets

```
df = pd.read_csv('../input/facial-expression-recognitionferchallenge/fer2013/fer2013/fer2013.csv')
print(df.shape)
df.head()
```

```
(35887, 3)
```

	emotion	pixels	Usage
0	0	70 80 82 72 58 58 60 63 54 58 60 48 89 115 121...	Training
1	0	151 150 147 155 148 133 111 140 170 174 182 15...	Training
2	2	231 212 156 164 174 138 161 173 182 200 106 38...	Training
3	4	24 32 36 30 32 23 19 20 30 41 21 22 32 34 21 1...	Training
4	6	4 0 0 0 0 0 0 0 0 0 0 0 0 3 15 23 28 48 50 58 84...	Training

```
df.emotion.unique()
```

```
array([0, 2, 4, 6, 3, 5, 1])
```

```
df = pd .read_csv('/kaggle/input/emotion-analysis-based-on-text/emotion_sentimen_dataset.csv', encoding=
```

```
df.head()
```

	Unnamed: 0	text	Emotion
0	0	i seriously hate one subject to death but now ...	hate
1	1	im so full of life i feel appalled	neutral
2	2	i sit here to write i start to dig out my feel...	neutral
3	3	ive been really angry with r and i feel like a...	anger
4	4	i feel suspicious if there is no one outside l...	neutral

- Data Cleaning

```python
# Function to expand contractions
def expand_contractions(text):
    contractions_dict = {
        "can't": "cannot", "won't": "will not", "n't": " not",
        "'re": " are", "'s": " is", "'d": " would", "'ll": " will",
        "'t": " not", "'ve": " have", "'m": " am"
    }
    for key, value in contractions_dict.items():
        text = re.sub(r"\b" + re.escape(key) + r"\b", value, text)
    return text

def clean_text(text):
    text = expand_contractions(text)  # Expand contractions
    text = text.lower()  # Convert to lowercase
    text = re.sub(r'@\w+', '', text)  # Remove @usernames
    text = re.sub(r'http\S+|www\S+', '', text)  # Remove URLs
    text = text.encode('ascii', 'ignore').decode('ascii')  # Remove emojis
    text = text.translate(str.maketrans('', '', string.punctuation))  # Remove punctuation
    text = " ".join([lemmatize_word(word) for word in text.split() if word not in stop_words])  # Lemmatization & Stopword removal
    text = re.sub(r'\s+', ' ', text).strip()  # Remove extra spaces
    return text


# Example usage
sample_text = "I'm happy! Can't wait to see you."
print(clean_text(sample_text))
```

```
happy can not wait to see
```

```python
# Apply preprocessing
df['Clean_Text'] = df['text'].astype(str).apply(clean_text)
```

- Regenerating Pixelated image

```python
fig = pyplot.figure(1, (14, 14))

k = 0
for label in sorted(df.emotion.unique()):
    for j in range(7):
        px = df[df.emotion==label].pixels.iloc[k]
        px = np.array(px.split(' ')).reshape(48, 48).astype('float32')

        k += 1
        ax = pyplot.subplot(7, 7, k)
        ax.imshow(px, cmap='gray')
        ax.set_xticks([])
        ax.set_yticks([])
        ax.set_title(emotion_label_to_text[label])
        pyplot.tight_layout()
```

Algorithms to be used:

- **Support Vector Machine (SVM)** – A supervised learning algorithm that finds the optimal hyperplane for classification. It works well for

high-dimensional data and is often used for text classification tasks like emotion detection.

```
from sklearn.svm import SVC

# Define SVM model
svm = SVC(kernel='linear', probability=True)

# Define TF-IDF + SVM pipeline
pipe_svm = Pipeline([
    ('tfidf', TfidfVectorizer(ngram_range=(1, 1), max_features=500)),
    ('svm', svm)
])
```

- **Logistic Regression** – A probabilistic model that predicts the probability of an instance belonging to a particular class. Despite its simplicity, it is effective in binary and multi-class classification problems.

```
# Build Logistic Regression Model with TfidfVectorizer
pipe_lr = Pipeline([
    ('tfidf', TfidfVectorizer(ngram_range=(1,1), max_features=500)),
    ('lr', LogisticRegression(max_iter=500, solver='liblinear'))
])
```

```
X_train_flat = X_train.reshape(X_train.shape[0], -1)
X_valid_flat = X_valid.reshape(X_valid.shape[0], -1)
```

```
y_train_labels = np.argmax(y_train, axis=1)
y_valid_labels = np.argmax(y_valid, axis=1)
```

```
# --- Logistic Regression ---
import time
print("\nTraining Logistic Regression...")
lr = LogisticRegression(max_iter=200, solver='saga', multi_class='multinomial', n_jobs=-1)
start = time.time()
lr.fit(X_train_flat, y_train_labels)
print(f"Training Time: {time.time() - start:.2f} seconds")
lr_preds = lr.predict(X_valid_flat)
print("Logistic Regression Accuracy:", accuracy_score(y_valid_labels, lr_preds))
print(classification_report(y_valid_labels, lr_preds))
```

- **Random Forest** – An ensemble learning method that builds multiple decision trees and merges their results to improve accuracy and reduce overfitting.

```
# --- Random Forest ---
print("\nTraining Random Forest...")
rf = RandomForestClassifier(n_estimators=100, n_jobs=-1, random_state=42)
start = time.time()
rf.fit(X_train_flat, y_train_labels)
print(f"Training Time: {time.time() - start:.2f} seconds")
rf_preds = rf.predict(X_valid_flat)
print("Random Forest Accuracy:", accuracy_score(y_valid_labels, rf_preds))
print(classification_report(y_valid_labels, rf_preds))
```

- **Naïve Bayes** – A probabilistic classifier based on Bayes' theorem, often used for text classification due to its efficiency in handling large feature spaces.

```
from sklearn.naive_bayes import MultinomialNB
from sklearn.pipeline import Pipeline
from sklearn.feature_extraction.text import TfidfVectorizer

# Define Naïve Bayes model
nb = MultinomialNB()

# Define TF-IDF + Naïve Bayes pipeline
pipe_nb = Pipeline([
    ('tfidf', TfidfVectorizer(ngram_range=(1, 1), max_features=500)),
    ('nb', nb)
])
```

- **XGBoost -** XGBoost (eXtreme Gradient Boosting) is a powerful and versatile machine learning algorithm used for both classification and regression tasks. It's an optimized implementation of gradient boosting, known for its speed, efficiency, and ability to handle large datasets.

```python
from xgboost import XGBClassifier

# Define XGBoost model
xgb = XGBClassifier(use_label_encoder=False, eval_metric='mlogloss')

# Define TF-IDF + XGBoost pipeline
pipe_xgb = Pipeline([
    ('tfidf', TfidfVectorizer(ngram_range=(1, 1), max_features=500)),
    ('xgb', xgb)
])
```

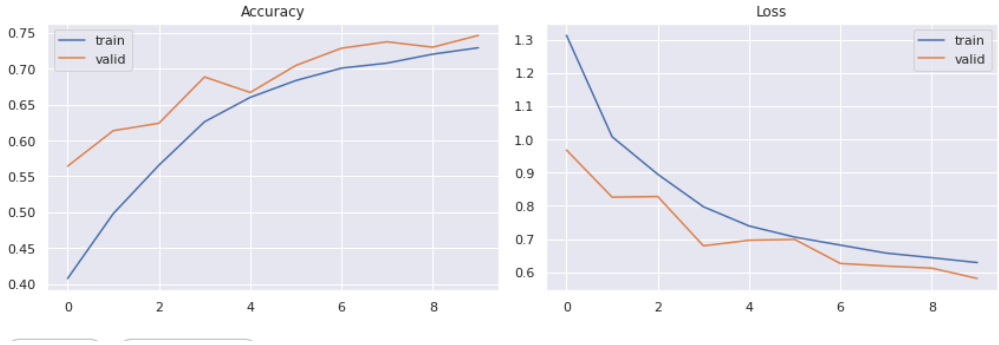
```python
# --- XGBoost ---
print("\nTraining XGBoost...")
xgb_model = XGBClassifier(use_label_encoder=False, eval_metric='mlogloss', n_jobs=-1, random_state=42)
start = time.time()
xgb_model.fit(X_train_flat, y_train_labels)
print(f"Training Time: {time.time() - start:.2f} seconds")
xgb_preds = xgb_model.predict(X_valid_flat)
print("XGBoost Accuracy:", accuracy_score(y_valid_labels, xgb_preds))
print(classification_report(y_valid_labels, xgb_preds))
```

- **TF-IDF -** TF-IDF, short for Term Frequency-Inverse Document Frequency, is a numerical statistic used to evaluate the importance of a word in a document relative to a collection of documents (corpus). It combines two concepts: Term Frequency (TF) and Inverse Document Frequency (IDF).

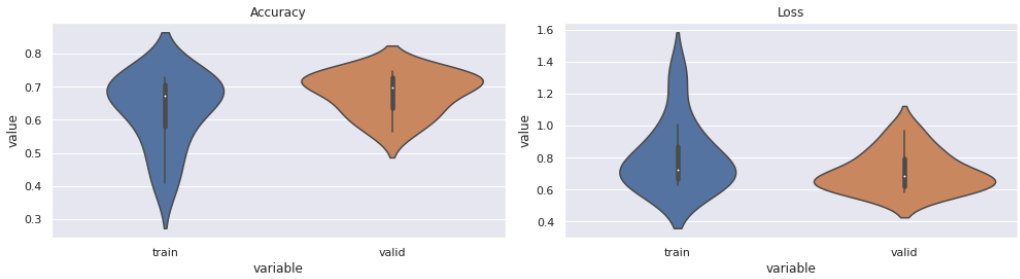
RESULTS

Data Visualization:

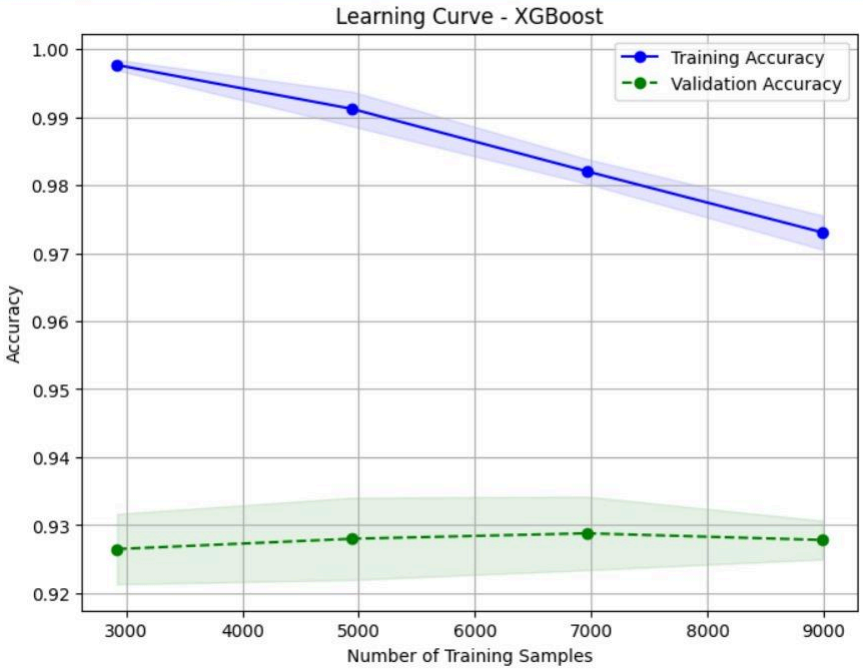
```
sns.set()
fig = pyplot.figure(0, (12, 4))

ax = pyplot.subplot(1, 2, 1)
sns.lineplot(history.epoch, history.history['accuracy'], label='train')
sns.lineplot(history.epoch, history.history['val_accuracy'], label='valid')
pyplot.title('Accuracy')
pyplot.tight_layout()

ax = pyplot.subplot(1, 2, 2)
sns.lineplot(history.epoch, history.history['loss'], label='train')
sns.lineplot(history.epoch, history.history['val_loss'], label='valid')
pyplot.title('Loss')
pyplot.tight_layout()

pyplot.savefig('epoch_history_dcnn.png')
pyplot.show()
```




```
df_accu = pd.DataFrame({'train': history.history['accuracy'], 'valid': history.history['val_accuracy']})
df_loss = pd.DataFrame({'train': history.history['loss'], 'valid': history.history['val_loss']})

fig = pyplot.figure(0, (14, 4))
ax = pyplot.subplot(1, 2, 1)
sns.violinplot(x="variable", y="value", data=pd.melt(df_accu), showfliers=False)
pyplot.title('Accuracy')
pyplot.tight_layout()

ax = pyplot.subplot(1, 2, 2)
sns.violinplot(x="variable", y="value", data=pd.melt(df_loss), showfliers=False)
pyplot.title('Loss')
pyplot.tight_layout()

pyplot.savefig('performance_dist.png')
pyplot.show()
```



```
warnings.warn(some_fits_failed_message, FitFailedWarning)
```

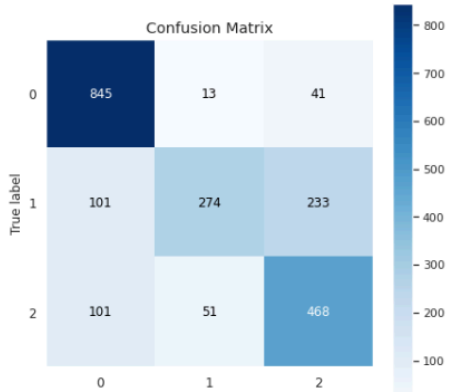


Accuracy and Precision of best evaluated model:

```
yhat_valid = model.predict_classes(X_valid)
scikitplot.metrics.plot_confusion_matrix(np.argmax(y_valid, axis=1), yhat_valid, figsize=(7,7))
pyplot.savefig("confusion_matrix_dcnn.png")

print(f'total wrong validation predictions: {np.sum(np.argmax(y_valid, axis=1) != yhat_valid)}\n\n')
print(classification_report(np.argmax(y_valid, axis=1), yhat_valid))
```

```
total wrong validation predictions: 540
```

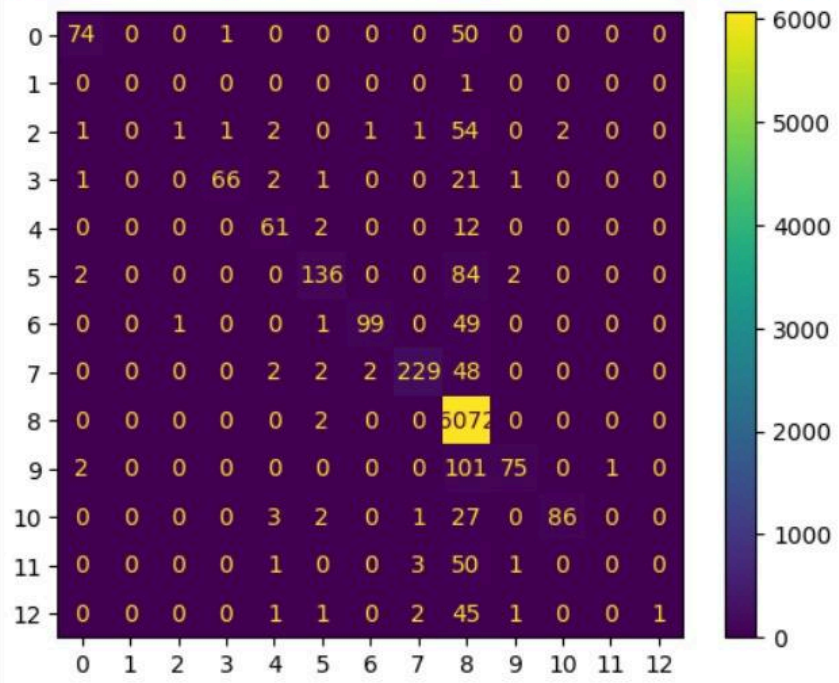
	precision	recall	f1-score	support
0	0.81	0.94	0.87	899
1	0.81	0.45	0.58	608
2	0.63	0.75	0.69	620
accuracy			0.75	2127
macro avg	0.75	0.72	0.71	2127
weighted avg	0.76	0.75	0.73	2127



[47]:
```
from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay

# Existing evaluation
acc_xgb = accuracy_score(y_test, y_pred_xgb)
print("\n=== XGBoost Results ===")
print(f"Accuracy: {acc_xgb:.4f}")
print("Classification Report:")
print(classification_report(y_test, y_pred_xgb))

# 📊 Confusion Matrix
cm = confusion_matrix(y_test, y_pred_xgb)
disp = ConfusionMatrixDisplay(confusion_matrix=cm, display_labels=le.classes_)
disp.plot(xticks_rotation=45)
```

```
=== XGBoost Results ===
Accuracy: 0.9211
Classification Report:
              precision    recall  f1-score   support

           0       0.93      0.59      0.72       125
           1       0.00      0.00      0.00         1
           2       0.50      0.02      0.03        63
           3       0.97      0.72      0.82        92
           4       0.85      0.81      0.83        75
           5       0.93      0.61      0.73       224
           6       0.97      0.66      0.79       150
           7       0.97      0.81      0.88       283
           8       0.92      1.00      0.96      6074
           9       0.94      0.42      0.58       179
          10       0.98      0.72      0.83       119
          11       0.00      0.00      0.00        55
          12       1.00      0.02      0.04        51

    accuracy                           0.92      7491
   macro avg       0.76      0.49      0.55      7491
weighted avg       0.91      0.92      0.91      7491
```

ValueError: The number of FixedLocator locations (13), usually from a call to



CONCLUSION

Our project IMAGE AND TEXT EMOTION DETECTION focuses on identifying human emotions from both visual and textual data. The primary objective of this project is to develop an intelligent system capable of accurately detecting and classifying emotions expressed through facial expressions and written text.

To achieve robust and reliable emotion recognition, we implemented and evaluated multiple machine learning approaches. For text-based emotion detection, models like Logistic Regression, Random Forest, XGBoost, and Naïve Bayes were employed using TF-IDF vectorization. For image-based emotion detection, trained on a dataset to classify emotion through text and facial expressions.

By integrating both text and image modalities and applying these advanced techniques, the project aims to contribute towards more emotionally-aware systems, enhancing applications in human-computer interaction, mental health monitoring, and personalized user experiences**.**

REFERENCES

1. C. Cortes and V. Vapnik, “Support-Vector Networks,” *Machine Learning*, vol. 20, no. 3, pp. 273–297, 1995.
2. D. W. Hosmer and S. Lemeshow, *Applied Logistic Regression*, 2nd ed., John Wiley & Sons, 2000.
3. L. Breiman, “Random Forests,” *Machine Learning*, vol. 45, no. 1, pp. 5–32, 2001.
4. Y. Freund and R. E. Schapire, “Experiments with a New Boosting Algorithm,” in *Proc. of the 13th International Conference on Machine Learning*, 1996, pp. 148–156.
5. J. Friedman, T. Hastie, and R. Tibshirani, “Additive Logistic Regression: A Statistical View of Boosting,” *Annals of Statistics*, vol. 28, no. 2, pp. 337–407, 2000.
6. I. Guyon et al., “Deep Learning for Facial Expression Recognition,” in *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, 2015.
7. T. E. Oliphant, “NumPy: A Guide to NumPy,” *Trelgol Publishing*, 2006.
8. W. McKinney, *Python for Data Analysis: Data Wrangling with Pandas, NumPy, and IPython*, 2nd ed., O’Reilly Media, 2017.
9. J. D. Hunter, “Matplotlib: A 2D Graphics Environment,” *Computing in Science & Engineering*, vol. 9, no. 3, pp. 90–95, 2007.
10. M. Waskom et al., “Seaborn: Statistical Data Visualization,” *Journal of Open Source Software*, vol. 6, no. 60, p. 3021, 2021.
11. S. van Rossum and F. L. Drake Jr., *Python Reference Manual*, Python Institute, 2003.
12. A. Goodman and D. Flaxman, “European Union Regulations on Algorithmic Decision-Making and a ‘Right to Explanation’,” *AI Magazine*, vol. 38, no. 3, pp. 50–57, 2017.
13. B. Mittelstadt, “Principles Alone Cannot Guarantee Ethical AI,” *Nature Machine Intelligence*, vol. 1, pp. 501–507, 2019.