RAJALAKSHMI EDUVERSE

20-02-2025

Instructor - Gowthami Shyam

Presented by Keerthana Praveen

DATA SCIENCE PROJECT

Deep Learning - CNN

Instructed by - Mrs.Gowthami shyam

Presented by - Keerthana Praveen

Acknowledgement

I would like to express my sincere gratitude to everyone who contributed to the successful completion of this project on Emotion Recognition using Deep Learning.

Firstly, I extend my heartfelt thanks to my instructor and mentors for their valuable guidance, insightful discussions, and continuous support throughout the project. Their expertise has been instrumental in shaping my understanding of deep learning concepts and their application to real-world problems.

This project has been a great learning experience, helping me apply theoretical knowledge to practical applications in the field of emotion recognition and computer vision.
Thank you.

TABLE
of contents

Introduction

Problem Statement

The task of emotion recognition involves detecting and classifying human emotions from facial expressions. In this project, we aim to build a deep learning model that can recognize emotions from images of human faces. The model will be trained on a dataset of facial images, where each image is labeled with one of eight emotions: happy, sad, angry, surprised, contempt, disgust, fear, or neutral.

Motivation

Emotion recognition is a critical task in areas such as human-computer interaction, healthcare, and security. For instance, in customer service, emotion recognition can help assess user satisfaction and personalize interactions. Additionally, in healthcare, recognizing emotions from facial expressions can assist in monitoring patients' psychological states. By leveraging deep learning, we can create a more accurate system that can automatically detect these emotions from images, providing better and more efficient solutions in these domains.

Objectives

1 Model Evaluation:

To evaluate the model's performance, we will use key metrics such as accuracy, precision, recall, and F1-score. In addition, a confusion matrix will be utilized to visualize the true versus predicted emotions, allowing for a deeper understanding of where the model excels and where it faces challenges. These metrics will help us identify which emotions are most accurately predicted and which require further refinement.

2 Insight Generation:

By analyzing the results and evaluating the confusion matrix, we aim to uncover insights into which facial features (such as specific facial expressions or areas of the face) contribute the most to accurate emotion prediction. These insights could lead to improvements in model performance and help refine the feature selection or data preprocessing steps in future work. Additionally, understanding which emotions are harder to classify will provide a clearer direction for further optimization of the model.

This gives a more detailed view of what you're trying to achieve, emphasizing the importance of evaluation and learning from model results to improve overall accuracy and robustness.

Data Description

Data Sources

The dataset used in this project is a publicly available Emotion Recognition dataset, typically sourced from Kaggle or similar repositories. This dataset contains images of faces with corresponding emotion labels. It is commonly used for training deep learning models on facial emotion recognition tasks.

Data Characteristics

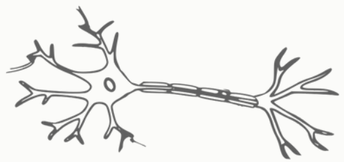
- Number of Samples: The dataset contains a large number of facial images, usually in the range of thousands.
- Features: Each image is 48x48 pixels in grayscale, representing a single facial expression.
- Emotions: The labels for the images correspond to one of eight emotions: happy, sad, angry, surprised, contempt, disgust, fear, and neutral.
- Data Format: The dataset is usually divided into training and testing sets, with images stored in arrays and labels encoded as categorical variables.

Methodology

Phase 1 - Apporach

- Data Preprocessing: The dataset is preprocessed by reshaping the images into a consistent format (48x48 pixels), scaling pixel values, and encoding the emotion labels. The training and testing datasets are also divided for validation.
- Model Selection and Training: A deep learning model (likely a Convolutional Neural Network - CNN) is built and trained on the facial emotion dataset using TensorFlow or Keras.
- Evaluation: After training the model, it is evaluated on the test set, where predictions are compared with true labels to compute the accuracy and other performance metrics.

Phase 2 - Architecture Used

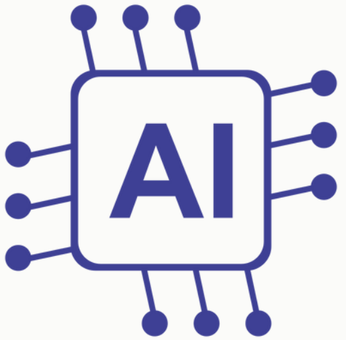


- For this project, the Convolutional Neural Network (CNN) is chosen due to its effectiveness in image classification tasks. CNNs excel at capturing spatial hierarchies in images and are ideal for tasks like emotion recognition.
- Model Architecture: The CNN model consists of convolutional layers followed by pooling layers, dense layers, and a softmax output layer to classify the emotions.
- Activation Function: ReLU (Rectified Linear Unit) is used for hidden layers, and Softmax is used for the output layer to predict probabilities for each emotion class.
- Loss Function: Categorical Cross-Entropy is used as the loss function since the problem is a multi-class classification task.

Evaluation Metrics

The following metrics are used to evaluate the performance of the emotion recognition model:

- Accuracy: Measures the overall correctness of the model by calculating the percentage of correctly classified emotions.
-
- Confusion Matrix: Helps visualize the classification performance across different emotion classes and understand misclassifications.
-
- Precision, Recall, F1-Score: These metrics provide insights into the model's performance per class, especially in imbalanced datasets.



Expected Outcomes

- High Accuracy: The deep learning model is expected to achieve high accuracy in predicting emotions from facial expressions.
- Clear Insights: The model will provide insights into which emotions are more difficult to predict and which facial features contribute most to classification.
- Improved Predictions: The use of deep learning, particularly CNNs, should lead to better predictions compared to traditional machine learning models.

Conclusion

In conclusion, this deep learning project on emotion recognition aims to tackle the challenge of accurately identifying human emotions from facial expressions using a convolutional neural network (CNN). Through the application of advanced neural network techniques, data preprocessing, and feature extraction, we seek to improve the model's ability to distinguish between various emotions such as happiness, sadness, anger, surprise, contempt, disgust, fear, and neutrality. The primary outcome is to achieve high accuracy and robustness in emotion recognition across different individuals and situations, ensuring that the model generalizes well to new, unseen data. By utilizing evaluation metrics like accuracy, confusion matrix, and visual comparisons of predicted versus true labels, we aim to assess the model's strengths and weaknesses. Furthermore, by gaining insights into which features of facial expressions are most influential in predicting emotions, we can provide valuable feedback for both refining the model and advancing the field of emotion recognition. This project also lays the groundwork for future applications, such as integrating emotion recognition systems into AI-driven platforms for healthcare, marketing, and human-computer interaction, where understanding human emotions plays a crucial role. Ultimately, the project not only advances the understanding of emotion detection but also provides a practical solution to a pressing real-world problem

Code

```
[1]: import os
     print(os.getcwd())  # This will show your current working directory

C:\Users\Lenova\Documents\data science\Pandas_datasets

[4]: import os

     # Print out the folder structure to check the paths
     image_folder = 'images'
     print(os.listdir(image_folder))  # List folders like 'happy', 'sad', etc.

['0', '1', '10', '11', '12', '13', '14', '15', '16', '17', '18', '2', '3', '4', '5', '6', '7', '8', '9']

[5]: label_mapping = {
         '0': 'happy',
         '1': 'sad',
         '2': 'angry',
         '3': 'surprised',
         '4': 'contempt',
         '5': 'disgust',
         '6': 'fear',
         '7': 'neutral'
     }
```

```
[7]: import os
     import cv2
     import numpy as np

     def load_images(image_folder, label_mapping, img_size=(48, 48)):
         images = []
         labels_list = []

         for folder_name in os.listdir(image_folder):
             label_path = os.path.join(image_folder, folder_name)

             if os.path.isdir(label_path) and folder_name in label_mapping:  # Ensure it's a valid folder
                 label = label_mapping[folder_name]  # Get mapped emotion label

                 for img_name in os.listdir(label_path):
                     img_path = os.path.join(label_path, img_name)
                     img = cv2.imread(img_path)

                     if img is not None:  # Ensure the image was loaded
                         img = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)  # Convert to grayscale
                         img = cv2.resize(img, img_size)  # Resize image
                         images.append(img)
                         labels_list.append(label)

         images = np.array(images)
         labels_list = np.array(labels_list)

         # Normalize the images
         images = images / 255.0
         images = images.reshape(-1, img_size[0], img_size[1], 1)  # Reshape for CNN input
         return images, labels_list

[8]: images, labels = load_images("images", label_mapping)
     print("Loaded images:", images.shape)
     print("Loaded labels:", labels.shape)

Loaded images: (64, 48, 48, 1)
Loaded labels: (64,)

[9]: from sklearn.preprocessing import LabelEncoder

     # Encode labels
     label_encoder = LabelEncoder()
     encoded_labels = label_encoder.fit_transform(labels)

     # Print mapping
     print("Label Encoding Mapping:", dict(zip(label_encoder.classes_, label_encoder.transform(label_encoder.classes_))))

Label Encoding Mapping: {'angry': 0, 'contempt': 1, 'disgust': 2, 'fear': 3, 'happy': 4, 'neutral': 5, 'sad': 6, 'surprised': 7}
```

Code

[10]:
```
from tensorflow.keras.utils import to_categorical

num_classes = len(label_encoder.classes_)  # Number of unique emotions
one_hot_labels = to_categorical(encoded_labels, num_classes)

print("One-hot encoded labels shape:", one_hot_labels.shape)
```

```
One-hot encoded labels shape: (64, 8)
```

[11]:
```
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv2D, MaxPooling2D, Flatten, Dense, Dropout

# Build CNN Model
model = Sequential([
    Conv2D(32, (3, 3), activation='relu', input_shape=(48, 48, 1)),
    MaxPooling2D((2, 2)),

    Conv2D(64, (3, 3), activation='relu'),
    MaxPooling2D((2, 2)),

    Flatten(),
    Dense(128, activation='relu'),
    Dropout(0.5),  # Helps prevent overfitting
    Dense(num_classes, activation='softmax')  # Output Layer
])

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

# Print model summary
model.summary()
```

```
C:\Users\Lenova\anaconda3\anaconda\Lib\site-packages\keras\src\layers\convolutional\base_conv.py:107: UserWarning: Do not pass an `input_shape`/`input_dim` argument to a layer. When using Sequential models, prefer using an `Input(shape)` object as the first layer in the model instead.
  super().__init__(activity_regularizer=activity_regularizer, **kwargs)
Model: "sequential"
```

Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 46, 46, 32)	320
max_pooling2d (MaxPooling2D)	(None, 23, 23, 32)	0
conv2d_1 (Conv2D)	(None, 21, 21, 64)	18,496
max_pooling2d_1 (MaxPooling2D)	(None, 10, 10, 64)	0
flatten (Flatten)	(None, 6400)	0
dense (Dense)	(None, 128)	819,328
dropout (Dropout)	(None, 128)	0
dense_1 (Dense)	(None, 8)	1,032

```
Total params: 839,176 (3.20 MB)
Trainable params: 839,176 (3.20 MB)
Non-trainable params: 0 (0.00 B)
```

[12]:
```
from sklearn.model_selection import train_test_split

# Split data
X_train, X_test, y_train, y_test = train_test_split(images, one_hot_labels, test_size=0.2, random_state=42)

# Train the model
model.fit(X_train, y_train, epochs=10, validation_data=(X_test, y_test), batch_size=32)
```

```
Epoch 1/10
2/2 ━━━━━━━━━━━━━━━━━━━━ 8s 1s/step - accuracy: 0.0758 - loss: 2.1090 - val_accuracy: 0.0769 - val_loss: 2.0094
Epoch 2/10
2/2 ━━━━━━━━━━━━━━━━━━━━ 0s 211ms/step - accuracy: 0.1646 - loss: 2.0332 - val_accuracy: 0.2308 - val_loss: 1.9831
Epoch 3/10
2/2 ━━━━━━━━━━━━━━━━━━━━ 0s 206ms/step - accuracy: 0.2324 - loss: 1.9887 - val_accuracy: 0.2308 - val_loss: 1.9228
Epoch 4/10
2/2 ━━━━━━━━━━━━━━━━━━━━ 0s 202ms/step - accuracy: 0.3002 - loss: 1.8914 - val_accuracy: 0.3077 - val_loss: 1.8232
Epoch 5/10
2/2 ━━━━━━━━━━━━━━━━━━━━ 0s 180ms/step - accuracy: 0.2559 - loss: 1.7952 - val_accuracy: 0.4615 - val_loss: 1.7090
Epoch 6/10
2/2 ━━━━━━━━━━━━━━━━━━━━ 0s 332ms/step - accuracy: 0.5065 - loss: 1.5930 - val_accuracy: 0.5385 - val_loss: 1.5589
Epoch 7/10
2/2 ━━━━━━━━━━━━━━━━━━━━ 0s 324ms/step - accuracy: 0.5039 - loss: 1.4698 - val_accuracy: 0.6923 - val_loss: 1.3652
Epoch 8/10
2/2 ━━━━━━━━━━━━━━━━━━━━ 0s 212ms/step - accuracy: 0.4257 - loss: 1.4869 - val_accuracy: 1.0000 - val_loss: 1.1558
Epoch 9/10
2/2 ━━━━━━━━━━━━━━━━━━━━ 0s 271ms/step - accuracy: 0.6371 - loss: 1.2960 - val_accuracy: 0.8462 - val_loss: 1.0042
Epoch 10/10
2/2 ━━━━━━━━━━━━━━━━━━━━ 0s 246ms/step - accuracy: 0.6789 - loss: 1.2561 - val_accuracy: 0.8462 - val_loss: 0.8736
```

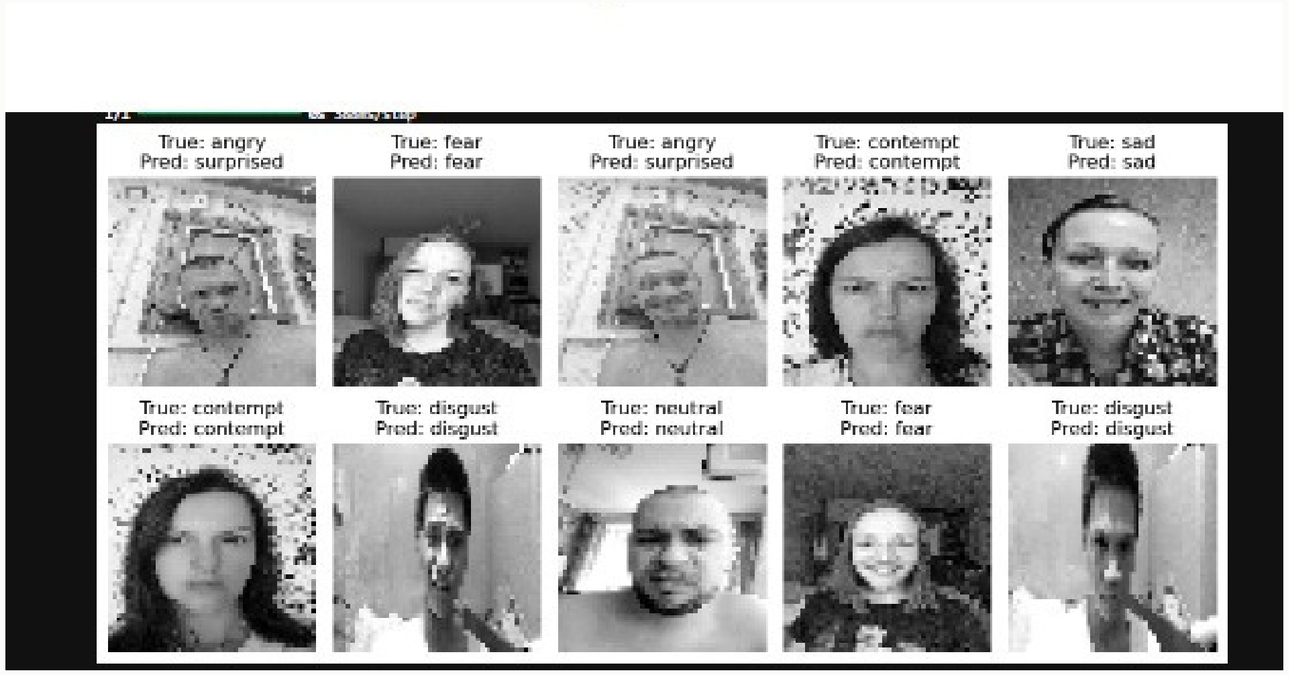
[12]:
```
<keras.src.callbacks.history.History at 0x1bbfd334d10>
```

[13]:
```
test_loss, test_acc = model.evaluate(X_test, y_test)
print("Test Accuracy:", test_acc)
```

```
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 146ms/step - accuracy: 0.8462 - loss: 0.8736
Test Accuracy: 0.8461538553237915
```

Code

```
[17]: import numpy as np
      import matplotlib.pyplot as plt

      # Get predictions for the full test set
      predictions = model.predict(X_test)

      # Convert predictions to class labels
      predicted_labels = np.argmax(predictions, axis=1)

      # Convert true labels to integer if needed
      true_labels = np.argmax(y_test, axis=1) if y_test.ndim > 1 else y_test

      # Define label mapping (ensure it's complete)
      label_mapping = {
          0: 'happy', 1: 'sad', 2: 'angry', 3: 'surprised',
          4: 'contempt', 5: 'disgust', 6: 'fear', 7: 'neutral'
      }

      # Select a few random images for comparison
      num_samples = 10
      indices = np.random.choice(len(X_test), num_samples, replace=False)

      plt.figure(figsize=(10, 5))

      for i, idx in enumerate(indices):
          img = X_test[idx].reshape(48, 48)  # Reshape for display
          true_emotion = label_mapping[true_labels[idx]]
          predicted_emotion = label_mapping[predicted_labels[idx]]

          plt.subplot(2, 5, i + 1)
          plt.imshow(img, cmap="gray")
          plt.title(f"True: {true_emotion}\nPred: {predicted_emotion}")
          plt.axis("off")

      plt.tight_layout()
      plt.show()
```



References

Rajalakshmi Eduverse Data Science Course – Mrs. Gowthami Shyam, Instructor
Course: Data Science
Institution: Rajalakshmi Eduverse
This course provided foundational knowledge and practical skills in data science, including data preprocessing, machine learning algorithms, deep learning and model evaluation.

Kaggle (2025). "Emotion Recognition Dataset."
Retrieved from https://www.kaggle.com/datasets
The dataset used for this project was sourced from Kaggle, which provided relevant data for training and evaluating the emotion recognition model.


Thank You