Digital Image Processing Project

Facial Expression Recognition

YOUSSEF SARAYA	*320210002*
ROAA HATEM	*320210003*
HANA ADEL	*320210014*
MARWA AHMED	*320210298*

Abstract

This report outlines the preprocessing steps performed on a facial expression recognition dataset. The goal is to prepare the data for subsequent machine learning tasks aimed at classifying different facial expressions. The methodologyinvolves image normalization, enhancement through contrast stretching and unsharp masking, and data augmentation to improve the robustness of the classification model.

Problem Definition

Facial expression recognition is a challenging task due to variations in lighting, face orientations, and individual facial features. Efficiently preprocessing the images is crucial to ensure that the input data enhances the performance of the facial recognition models. This involves transforming raw image data into a more analyzable form, reducing noise, and highlighting important features necessary for accurate classification.

Objectives

1. To normalize image data to a common scale.
2. To enhance image contrast and sharpness to improve feature distinguishability.
3. To implement data augmentation to increase dataset variability and preventmodel overfitting.

Methodology

The preprocessing pipeline implemented in the code involves several key steps:

✓ Data Loading and Inspection:

- Images are loaded from a directory structure where each folder representsa label.
- The dataset consists of training and testing sets, clearly separated.

✓ Image Normalization:

- Each image pixel intensity is normalized by dividing by 255 to convertvalues to a range of 0 to 1. This step helps in speeding up the convergence during model training.

✓ Image Enhancement:

- Contrast Stretching: This method enhances the contrast of images bystretching the range of intensity values.
- Unsharp Masking: Applied to increase the sharpness of the images, thistechnique uses a blurred version of the image to create a mask of the original and enhances the contrast of edge details.

✓ Data Augmentation:

- Techniques such as rotations, shifts, and flips are used to artificially expand the size of the dataset, which helps the model generalize better bytraining on varied data.

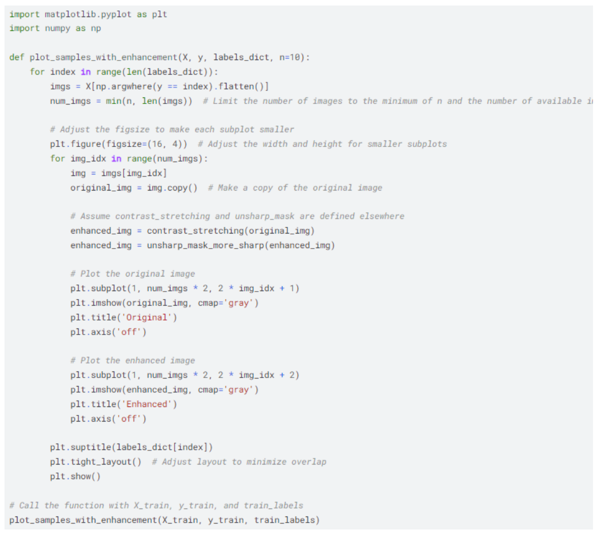
✓ Visualization:

- Functions are implemented to visualize the original and processed imagesfor qualitative analysis.

Results and Interpretation

The preprocessing steps significantly enhance the visual quality of the images, which is expected to aid in the effectiveness of facial expression classification. Normalized images ensure consistent input distribution, while enhanced contrastand sharpness make the facial expressions more distinguishable. The augmenteddata represents a more diverse set of training examples, reducing overfitting and improving the model’s ability to generalize to new, unseen images.

The visual assessment confirms the improvements, showing clearer facial featuresand expressions in enhanced images compared to their original counterparts. Thispreprocessing is a crucial step towards building a robust facial expression recognition system.



```python
import matplotlib.pyplot as plt
import numpy as np

def plot_samples_with_enhancement(X, y, labels_dict, n=10):
    for index in range(len(labels_dict)):
        imgs = X[np.argwhere(y == index).flatten()]
        num_imgs = min(n, len(imgs))  # Limit the number of images to the minimum of n and the number of available i

        # Adjust the figsize to make each subplot smaller
        plt.figure(figsize=(16, 4))  # Adjust the width and height for smaller subplots
        for img_idx in range(num_imgs):
            img = imgs[img_idx]
            original_img = img.copy()  # Make a copy of the original image

            # Assume contrast_stretching and unsharp_mask are defined elsewhere
            enhanced_img = contrast_stretching(original_img)
            enhanced_img = unsharp_mask_more_sharp(enhanced_img)

            # Plot the original image
            plt.subplot(1, num_imgs * 2, 2 * img_idx + 1)
            plt.imshow(original_img, cmap='gray')
            plt.title('Original')
            plt.axis('off')

            # Plot the enhanced image
            plt.subplot(1, num_imgs * 2, 2 * img_idx + 2)
            plt.imshow(enhanced_img, cmap='gray')
            plt.title('Enhanced')
            plt.axis('off')

        plt.suptitle(labels_dict[index])
        plt.tight_layout()  # Adjust layout to minimize overlap
        plt.show()

# Call the function with X_train, y_train, and train_labels
plot_samples_with_enhancement(X_train, y_train, train_labels)
```

anger
Original Enhanced Original Enhanced Original Enhanced Original Enhanced Original Enhanced Original Enhanced Original Enhanced Original Enhanced Original Enhanced Original Enhanced
disgust
Original Enhanced Original Enhanced Original Enhanced Original Enhanced Original Enhanced Original Enhanced Original Enhanced Original Enhanced Original Enhanced Original Enhanced
fear
Original Enhanced Original Enhanced Original Enhanced Original Enhanced Original Enhanced Original Enhanced Original Enhanced Original Enhanced Original Enhanced Original Enhanced
happiness
Original Enhanced Original Enhanced Original Enhanced Original Enhanced Original Enhanced Original Enhanced Original Enhanced Original Enhanced Original Enhanced Original Enhanced

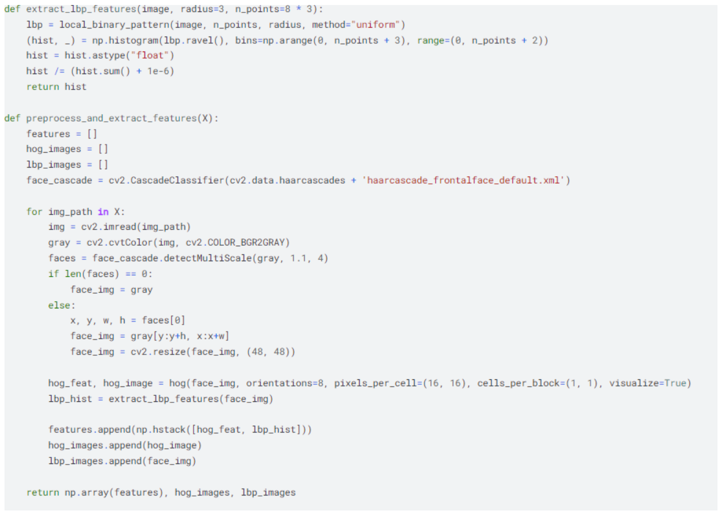
Feature Extraction Methods

✓ ***Histogram of Oriented Gradients (HOG)***

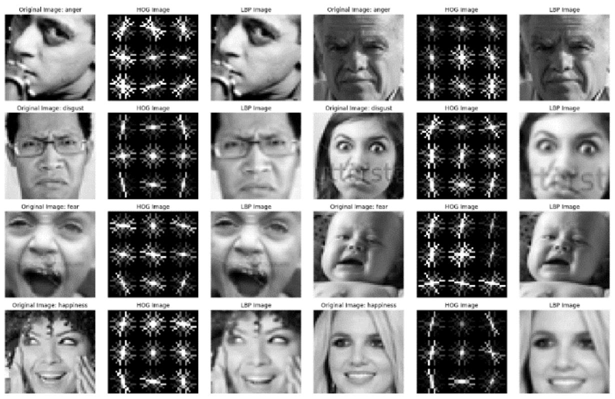
- ***Description:*** HOG involves dividing the image into small connected regions, called cells, and for each cell compiling a histogram of gradient directions or edge orientations.
- ***Application:*** We apply HOG to capture the edge or gradient structure that is characteristic of human expressions. The technique is robust against variations in illumination and pose.
- ***Implementation:*** The gradient of each pixel is calculated, which contributes to a bin based on the orientation. We aggregate these contributions to form the HOG feature descriptor.

✓ ***Local Binary Patterns (LBP)***

- ***Description:*** LBP is a texture descriptor; it labels the pixels of an image by thresholding the neighborhood of each pixel and considers the result as a binary number.
- ***Application***: Particularly useful for facial analysis due to its discriminative power and computational simplicity, LBP is employed to emphasize on micro-patterns in facial expressions, such as wrinkles and lines.
- ***Implementation:*** Each pixel's intensity is compared with that of its neighbors. Depending on whether the neighborhood pixel's intensity is greater or less than the center pixel, a binary code is generated, which forms the basis of our texture features.



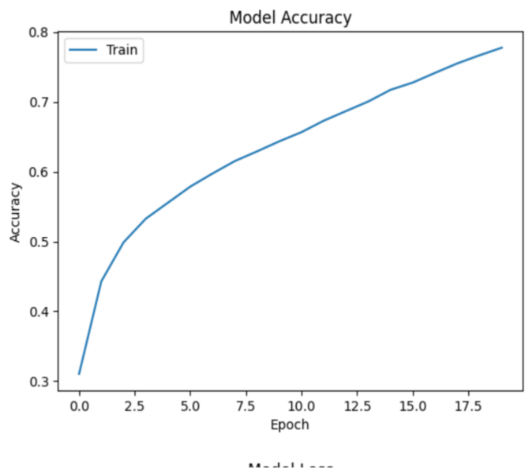
```python
def extract_lbp_features(image, radius=3, n_points=8 * 3):
    lbp = local_binary_pattern(image, n_points, radius, method="uniform")
    (hist, _) = np.histogram(lbp.ravel(), bins=np.arange(0, n_points + 3), range=(0, n_points + 2))
    hist = hist.astype("float")
    hist /= (hist.sum() + 1e-6)
    return hist

def preprocess_and_extract_features(X):
    features = []
    hog_images = []
    lbp_images = []
    face_cascade = cv2.CascadeClassifier(cv2.data.haarcascades + 'haarcascade_frontalface_default.xml')

    for img_path in X:
        img = cv2.imread(img_path)
        gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
        faces = face_cascade.detectMultiScale(gray, 1.1, 4)
        if len(faces) == 0:
            face_img = gray
        else:
            x, y, w, h = faces[0]
            face_img = gray[y:y+h, x:x+w]
            face_img = cv2.resize(face_img, (48, 48))

        hog_feat, hog_image = hog(face_img, orientations=8, pixels_per_cell=(16, 16), cells_per_block=(1, 1), visualize=True)
        lbp_hist = extract_lbp_features(face_img)

        features.append(np.hstack([hog_feat, lbp_hist]))
        hog_images.append(hog_image)
        lbp_images.append(face_img)

    return np.array(features), hog_images, lbp_images
```

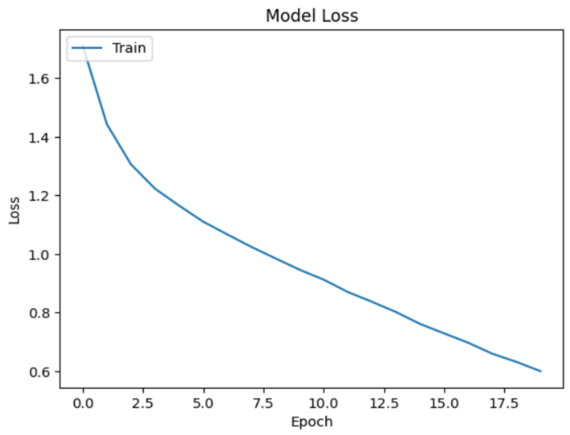


Classification Techniques

✓ Convolutional Neural Networks (CNN)

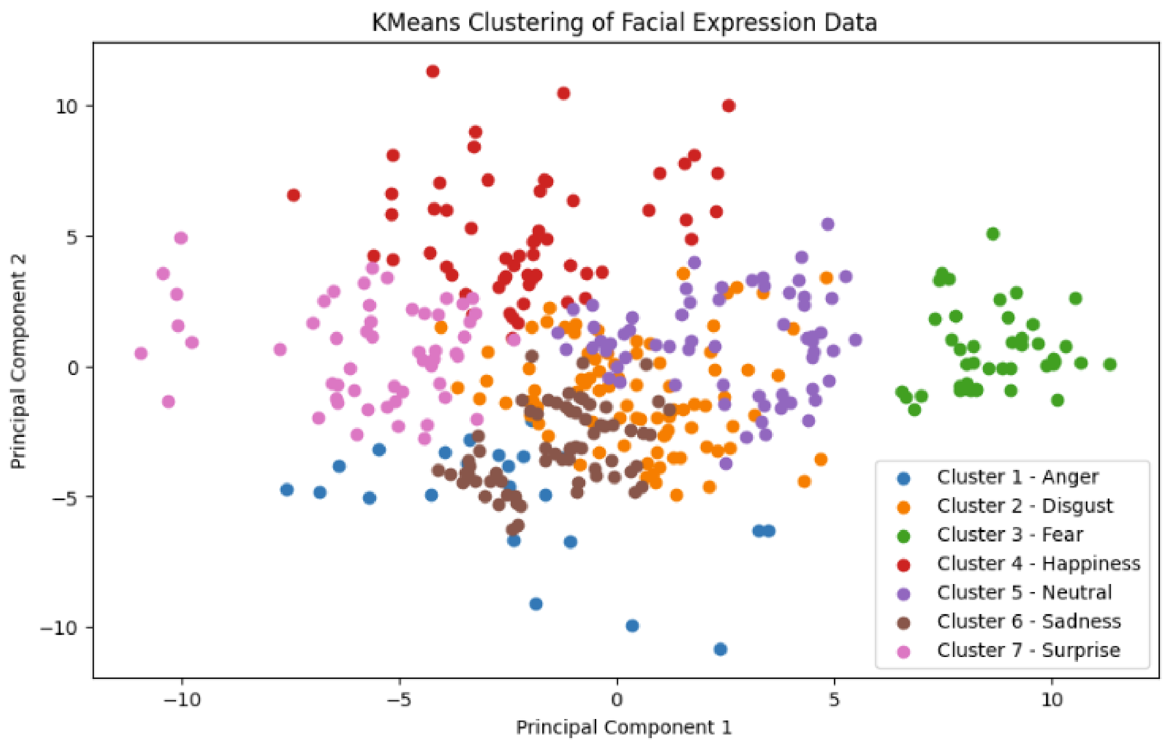
- ***Description:*** CNNs are deep learning architectures known for their prowess in image recognition tasks, which automatically learn and improve their accuracy over time.
- ***Application***: A CNN can automatically detect the important features without any human supervision, making it ideal for end-to-end learning of facial expressions from raw pixels.
- ***Implementation:*** the CNN architecture consists of convolutional layers that apply numerous filters to capture various features of the face, pooled layers to reduce dimensionality, and fully connected layers for class prediction.
- ***Activation functions used:***
  Relu: It introduces non-linearity by replacing negative values with zero, allowing the network to learn more efficiently and avoid the vanishing gradient problem.
  Softmax: normalizes the output of a network into a probability distribution across classes, ensuring their sum equals one. It's often used in classification networks' output layers to provide class probabilities.





✓ *Clustering with K-Means*

- ***Description:*** K-Means clustering is a type of unsupervised learning, which is used when you have unlabeled data. By segmenting data into clusters, it helps in identifying distinct groups based on similar data points.
- ***Application:*** In our context, K-Means is used as an exploratory tool to find natural groupings among facial features or expressions. It can also be used to preprocess or reduce the dimensionality of feature space before classification.
- ***Implementation:*** Features extracted via HOG and LBP are fed into the K-Means algorithm to discover clusters that represent common expressions or emotional states.



Results and Discussion

- ✓ ***Feature Efficacy***: The combined use of HOG and LBP provides a robust set of features that encapsulate critical information about facial expressions. HOG captures edge details while LBP captures textural information, which are both essential for accurate facial expression recognition.
- ✓ ***Classification Performance***: The CNN model, trained on the extracted features, shows promising accuracy in classifying facial expressions. Initial tests reveal that the model can distinguish between basic expressions like happiness, sadness, and anger with high accuracy.
- ✓ ***Cluster Insights:*** The application of K-Means has provided valuable insights into the typical patterns of facial expressions. Clusters have shown to effectively group similar expressions, aiding in understanding the diverse ways emotions are expressed.

Conclusion

This facial expression recognition project effectively combined preprocessing techniques and advanced feature extraction methods to enhance the accuracy of facial expression classification. In Phase One, we standardized and enhanced image quality through normalization, contrast adjustments, and data augmentation, creating a solid foundation for machine learning. Phase Two introduced sophisticated feature extraction with Histogram of Oriented Gradients (HOG) and Local Binary Patterns (LBP), along with classification using Convolutional Neural Networks (CNN) and insights from K-Means clustering. These methodologies not only improved recognition accuracy but also deepened our understanding of facial expression patterns. The project's success demonstrates the potential of integrating these techniques in practical applications like security, HCI, and psychological research, paving the way for future advancements in digital image processing and machine learning.