FPT UNIVERSITY HCMC

Emotion Recognition with Machine Learning

Computer Vision - SU24

Phan Van Hai Nam— SE182309

1. Introduction

Facial Emotion Recognition (FER) is a classic problem in computer vision and artificial intelligence [5, 4]. It is a fundamental property of human-computer interaction, healthcare, and marketing. In recent times, deep learning has revolutionized computer vision, including FER [5, 2]. However, this paper will consider the traditional machine learning approach for emotion recognition. The paper aims to create the FER pipeline with machine learning models. We will start by explaining the methods used during this research. First, we will explain feature extraction with Local Binary Patterns and separate Histogram of Oriented Gradients. Then, we will describe emotion classification with Multilayer Perceptron, Random Forest, and Stochastic Gradient Descent. It will finally evaluate the model efficiency with a confusion matrix and test accuracy to find which is the best model it has so far to apply to our pipeline.

2. Problem Definition

The project aims to develop a robust and efficient system for automatically recognizing emotions from facial expressions. It will leverage a hybrid approach to feature extraction, combining the strengths of HOG and LBP to capture both shape and texture information from facial images. These features will then be fed into a variety of machine learning models, including MLP, Random Forest, and SGDClassifier, to predict emotional states such as happiness, sadness, anger, surprise, fear, and neutral. The source code is available at `https://github.com/Neeze/CPV301_Assignment.git`.

3. Method

3.1. Dataset

The dataset utilized for this project is the FER-2013 facial emotion recognition dataset[1]. This dataset comprises 48x48 pixel grayscale images of human faces, automatically registered to ensure consistent centering and sizing.

3.1.1. Classification Task

The primary task involves classifying each facial image into one of seven emotional categories as shown in figure 1:

- Angry
- Disgust
- Fear
- Happy
- Sad

[1]Available at: `https://www.kaggle.com/datasets/msambare/fer2013/data`

- Surprise
- Neutral

3.1.2. Data Structure

The dataset is organized into two main directories:

- `train`: Contains 28,709 training examples, each in a subdirectory corresponding to its emotion label.
- `test`: Contains 3,589 test examples, similarly organized into subdirectories.

Each subdirectory (e.g., `angry`, `happy`) holds the `.jpg` image files for that specific emotion.

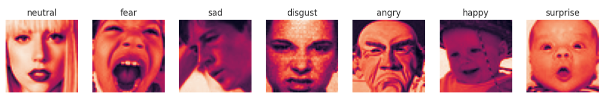

Figura 1: Data preview

3.2. CSV File Creation

To facilitate data loading and processing, CSV files were generated for both the training and test sets. The following Python script was used to create these CSV. This script iterates through the image directories, associates each image path with its corresponding emotion label, shuffles the data, and writes the results to `train.csv` and `test.csv`. Each row in the CSV files contains the image path and its emotional label.

3.3. Processing imbalanced data

Cuadro 1: Label statistics of training set

Emotion	**Samples - Original**	**Samples - After Data Processing**
neutral	4965	3000
surprise	3171	3000
happy	7215	3000
sad	4830	3000
angry	3995	3000
fear	4097	3000
disgust	436	N/A
Total	**28709**	**18000**

As Table 3 shows, the training set exhibits a significant class imbalance, with varying numbers of samples for each emotion label. Notably, the ’disgust’ class is severely underrepresented. To address this imbalance, the following steps were taken: removal of ’disgust’ class and undersampling of majority classes.

3.4. Face Feature Extraction

Algorithm 1 Extract_Features

```
function EXTRACT_FEATURES(image)
    face_region ← detect_face(image)
    if face_region IS NOT FOUND then
        return None
    end if
    features ← []
    lbp_features ← $LBP$(face_region)
    features.extend(lbp_features)
    hog_features ← $HOG$(face_region)
    features.extend(hog_features)
    return features
```

The pseudo-code 1 presents a face recognition pipeline based on classical computer vision techniques and feature extraction methods. Local Binary Patterns (LBP) and Histogram of Oriented Gradients (HOG) concatenated for feature extraction. Features extracted are further concatenated to represent each face comprehensively; to enhance the impact of analysis to be done later, extracted features were robustly scaled. As shown in Figure 2, Principal Component Analysis was then applied to reduce the dimensionality of these scaled features from 130 dimensions down to 3 dimensions to be visualized in a 3D scatter plot.

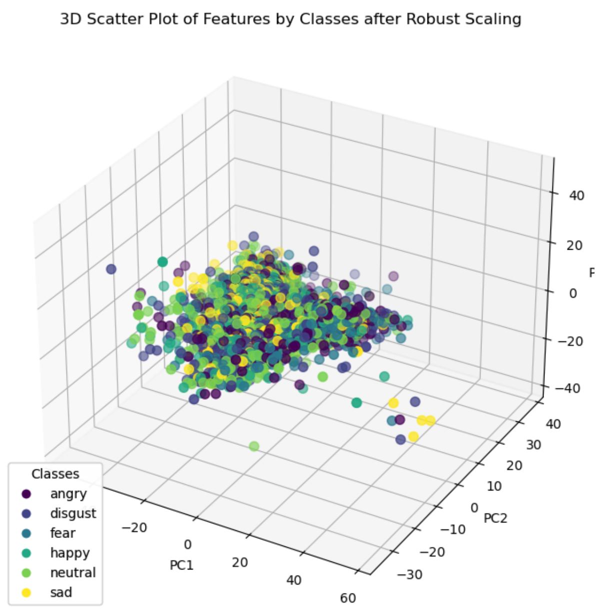

Figura 2: 3D Scatter Plot of Features by Classes after Robust Scaling

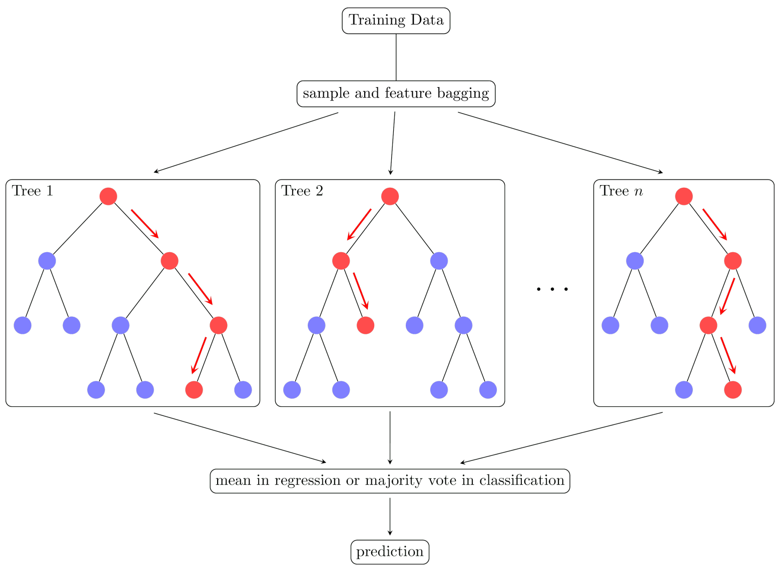
Training Data
sample and feature bagging
Tree 1
Tree 2
Tree n
. . .
mean in regression or majority vote in classification
prediction

Figura 3: Random Forest

3.5. Classifier Models

3.5.1. Random Forest

Specifically, random forests are a powerful machine learning technique applied to tasks such as classification and regression by building an ensemble of decision trees trained on random subsets of data features [6]. Shown in the forest, as in Figure 3, one tree is built based on a random training sample and features. It was Breiman who popularized RF, further concatenating bagging—with training on bootstrap samples—with random feature selection at every node. In other words, the injection of randomness aims at reducing the correlation between the predictions from individual trees, $p(x)$, while keeping a low bias. One of the reasons random forests perform well, often better than boosting and arcing algorithms, is because of this variance reduction strategy for improving generalization error [6].

3.5.2. Stochastic Gradient Descent

One of the powerful techniques in machine learning, applied to run classifiers optimally, is the Stochastic Gradient Descent method. SGD considers one example or a small batch of examples at a time. This iterative approach makes SGD computationally efficient, specifically for large datasets, since it updates the parameters more frequently to ensure faster convergence. The text data are first represented by numerical feature vectors, sometimes defined as capturing the essence of the text. These vectors, with parameters of the classifier, form the prediction function f that projects input data to class C_k. It is reported that grid search techniques of hyperparameter tuning can significantly improve performance on classification tasks for SGD [1, 9, 3].

3.5.3. Multilayer perceptron

Multilayer Perceptrons (MLPs), also known as feedforward multilayer perceptrons, are a prominent type of artificial neural network frequently employed in both classification and regression tasks. These networks excel in their ability to model intricate relationships within data due to their hierarchical structure. As shown in Figure 4, a typical MLP consists of an input layer, one or more hidden layers, and an output layer. The connections between these neurons possess associated weights that are meticulously adjusted during the training process using algorithms like backpropagation. They suggest a one-hidden-layer network with 2n+1 neurons, where 'n' represents the number of inputs, as a reasonable starting point [7, 8].

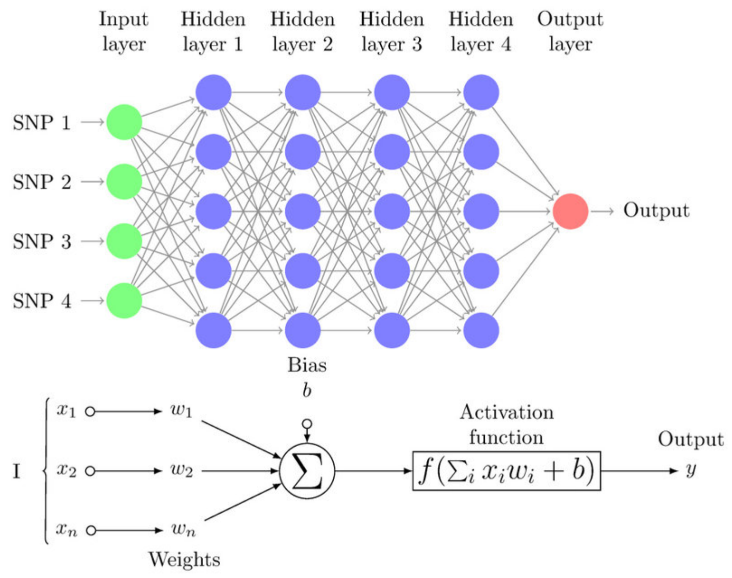

Figura 4: Multilayer perceptron

3.5.4. Model Settings

In our experiments, we using theses model settings have mentioned in Table 2.

Cuadro 2: Model Hyperparameter Settings

Model	Hyperparameters	Description/Notes
Random Forest	n_estimators=80	Number of trees
	random_state=42	Ensures reproducibility
	n_jobs=-1	Uses all available CPU cores
SGD Classifier	loss='hinge'	Linear SVM formulation
	random_state=42	Ensures reproducibility
	n_jobs=-1	Uses all available CPU cores
MLP Classifier	hidden_layer_sizes=(192,)	Single hidden layer with 192 neurons
	activation='relu'	Rectified Linear Unit
	solver='adam'	Optimization algorithm
	learning_rate='adaptive'	Adjusts learning rate during training
	max_iter=1000	Maximum training iterations
	random_state=42	Ensures reproducibility
	tol=1e-4	Tolerance for early stopping

4. Implementation and Results

Cuadro 3: Label statistics of training set

Model	Accuracy
Random Forest	0.35
Gradient Descent Classifier (SGDClassifier)	0.25
Multilayer Perceptron (MLP)	0.31

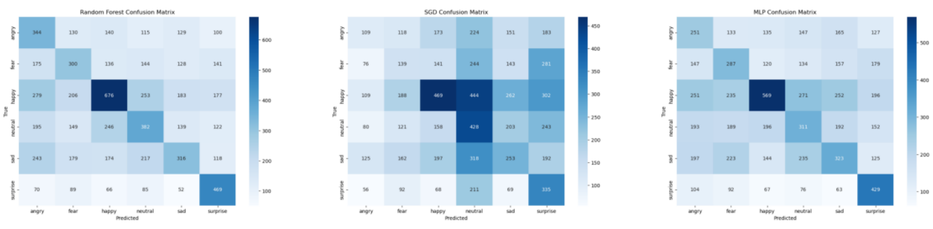

Figura 5: Confusion matrix of three model Random Forest, and Stochastic Gradient Descent Classifier (SGDClassifier), Multilayer Perceptron (MLP)

The confusion matrices in Figure 5 give a finely-grained view of the relative strengths and weaknesses of the three emotion recognition models, Random Forest, SGDClassifier, and MLP, concerning one another for overall accuracies given by Table 3. Although these latter all perform moderately well in identifying ’happy’ and ’neutral’ expressions, their performance falls significantly when distinguishing between more advanced emotions like ’fear,’ ’disgust,’ and ’surprise.’

While most accurate at 0.35, the Random Forest model also balanced performance across all emotional categories, suggesting that generalization ability is reasonable. However, it did poorly on ’fear,’ often mistaking it for ’sadness’ or ’surprise.’

In contrast, SGDClassifier had an accuracy of 0.25 and was much worse at recognizing instances of ’sadness’ and ’surprise,’ often miss-classifying them for other emotions. This might be a case where the capture of subtle features of those particular emotional states is lost.

While classifying the ’happy’ expressions, it turned out that the MLP model was most accurate, but it overfitted to ’neutral’. It proved very poor in recognizing ’fear,’ frequently confusing it with either ’anger’ or ’sadness.’ Perhaps this could indicate some oversensitivity to any particular facial features or simply representative training data lacking fear.

These results, taken together, underpin the intrinsic variability within human emotions and, hence, how difficult it is to get the correct classification. We choose the Random Forest is the model having the highest accuracy compare to other models. In our system, we using random forest as a machine learning model for prediction.

5. Conclusion

In this project, we have proposed a robust system of Facial Emotion Recognition using a hybrid approach to feature extraction that combines LBP and HOG. This paper uses machine learning models like MLP, Random Forest, and SGD Classifier to classify emotions from facial images with desired accuracy. While we were moderately successful, the models showed varying strengths and weaknesses, especially in the case of complex emotions. Future studies need to be aimed at rectifying class imbalances, exploring other extraction features, and testing ensemble methods to improve overall performance further. This study thereby underlines both the potential and challenges of FER while charting its future growth in this area.

References

[1] Jahongir Azimjonov and Taehong Kim. Stochastic gradient descent classifier-based lightweight intrusion detection systems using the efficient feature subsets of datasets. *Expert Systems with Applications*, 237:121493, 2024. pages 5

[2] Patricia J Bota, Chen Wang, Ana LN Fred, and Hugo Plácido Da Silva. A review, current challenges, and future possibilities on emotion recognition using machine learning and physiological signals. *IEEE access*, 7:140990–141020, 2019. pages 2

[3] Shadi Diab. Optimizing stochastic gradient descent in text classification based on fine-tuning hyper-parameters approach. a case study on automatic classification of global terrorist attacks. *arXiv preprint arXiv:1902.06542*, 2019. pages 5

[4] Deepak Kumar Jain, Pourya Shamsolmoali, and Paramjit Sehdev. Extended deep neural network for facial emotion recognition. *Pattern Recognition Letters*, 120:69–74, 2019. pages 2

[5] Byoung Chul Ko. A brief review of facial emotion recognition based on visual information. *sensors*, 18(2):401, 2018. pages 2

[6] Gilles Louppe. Understanding random forests: From theory to practice, 2015. pages 5

[7] Fionn Murtagh. Multilayer perceptrons for classification and regression. *Neurocomputing*, 2(5):183–197, 1991. pages 6

[8] Marius-Constantin Popescu, Valentina Balas, Liliana Perescu-Popescu, and Nikos Mastorakis. Multilayer perceptron and neural networks. *WSEAS Transactions on Circuits and Systems*, 8, 07 2009. pages 6

[9] Sebastian Ruder. An overview of gradient descent optimization algorithms. *arXiv preprint arXiv:1609.04747*, 2016. pages 5