# AI Detector *by SciSpace*

FPTUniv_Report.pdf • 8 July 2024
2260 words (16062 characters)

## Completely Human 0%

The text is entirely written by humans, without any assistance from AI.

| AI weightage | Content weightage | Sentences |
|---|---|---|
| H Highly AI written | **0%** Content | 0 |
| M Moderately AI written | **0%** Content | 0 |
| L Lowly AI written | **0%** Content | 0 |


FPT UNIVERSITY HCMC

# Emotion Recognition with Machine Learning

Computer Vision - SU24

Phan Van Hai Nam— SE182309

# 1. Introduction

Facial Emotion Recognition (FER), a key research area within computer vision and artificial intelligence, focuses on enabling computers to automatically recognize and interpret human emotions from facial expressions. [6] [5] With significant academic and commercial potential, FER plays a crucial role in various applications, including human-computer interaction, healthcare, and marketing.[6][2] Facial expressions, considered one of the primary channels of interpersonal communication, provide valuable insights into an individual's emotional state, making FER an essential aspect of understanding human behavior.[6]

In recent years, the advent of deep learning has revolutionized the field of computer vision, including FER. Deep learning models, particularly Convolutional Neural Networks (CNNs), have demonstrated remarkable capabilities in automatically learning discriminative features from raw image data, leading to substantial improvements in FER accuracy. Researchers have explored various deep learning architectures, including CNNs, Deep Belief Networks (DBNs), and Long Short-Term Memory (LSTM) networks, to enhance FER performance.[5][7]

However, in this article we approach the traditional machine learning method in recognizing emotions. This project delves into the world of FER, constructing a model designed to classify emotions based on facial images. To achieve this, we make the pipeline to extract face features such as: Local Binary Patterns (LBP) Features[10], Histogram of Oriented Gradients (HOG) Features[4]. And we are using these features to recognize the emotions.

Our approach involves extracting LBP features and HOG features images. By combining these two feature sets, we aim to create a comprehensive representation of facial expressions that captures both fine-grained details and overall facial structure. We then utilize a machine learning model such as Multilayer Perceptron (MLP), Random Forest, and Stochastic Gradient Descent Classifier (SGDClassifier) that trained on this combined feature set, and optimize its performance through hyperparameter tuning and cross-validation. To enhance efficiency when dealing with large datasets, we employ parallel processing techniques during feature extraction. Finally, we assess our model's effectiveness using a confusion matrix and test accuracy, offering valuable insights into its strengths and weaknesses.

# 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
- 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.

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

## 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** |

The training set, as illustrated in Table 2, exhibits a significant class imbalance, with varying numbers of samples for each emotion label. Notably, the 'disgust' class is severely underrepresented with only 436 samples. To address this imbalance and create a more suitable dataset for model training, the following steps were taken:
Removal of 'disgust' Class: Due to its extremely limited sample size, the entire 'disgust' class was removed from the dataset. This decision was made to prevent potential biases or inaccuracies that might arise from training a model on such a small and unrepresentative subset.
Undersampling of Majority Classes: To further mitigate the effects of class imbalance, a random undersampling strategy was employed for the remaining emotion classes. Specifically, for each class with more than 3000 samples ('surprise', 'neutral', 'happy', 'sad', 'angry', and 'fear'), the number of samples was reduced to approximately 3000. This was achieved through random selection, ensuring that the remaining samples for each class are representative of the original distribution.
By implementing these data preprocessing techniques, a more balanced dataset was created, facilitating a more effective and unbiased training process for the subsequent machine learning model.

## 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 outlines a face recognition pipeline that leverages a combination of classic computer vision techniques and feature extraction methods. For feature extraction, a combination of Local Binary Patterns (LBP), and Histogram of Oriented Gradients (HOG) were employed. The extracted features were concatenated to create a comprehensive representation of each face. To enhance the effectiveness of subsequent analysis, Robust Scaling was applied to all extracted features. This scaling method is particularly suitable when dealing with data that may contain outliers, as it mitigates the impact of extreme values by scaling features based on their interquartile range (IQR) rather than their standard deviation.
As depicted in Figure 2, Principal Component Analysis (PCA) was then utilized to reduce the dimensionality of the scaled features from 130 to 3, facilitating visualization in a 3D scatter plot. This visualization allows for the exploration of potential clusters or patterns within the data, which could be informative for downstream tasks such as face recognition or classification.

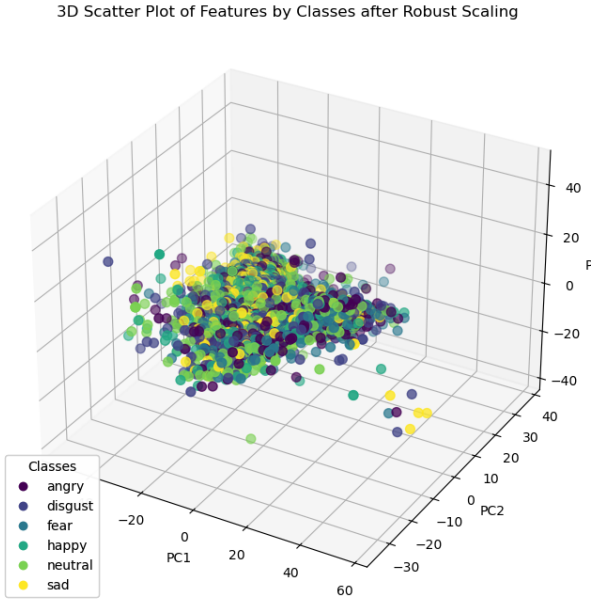


**Figura 2:** 3D Scatter Plot of Features by Classes after Robust Scaling

## 3.5. Classifier Models

### 3.5.1. Random Forest

Random forests, a robust machine learning method used for tasks like classification and regression, involve building an ensemble of decision trees trained on randomized data subsets [8]. As show in Figure 3 each tree in the forest is constructed using a random selection of training samples and features, promoting diversity among the individual learners and reducing the ensemble's overall variance. Breiman with popularizing random forests, combining bagging (training on bootstrap samples) with random feature selection at each node, resulting in a highly effective method applicable to a wide range of problems. The process of injecting randomness aims to minimize correlation between individual tree predictions $p(x)$ while maintaining low bias. Randomized Trees (ETs), a variant that further pushes randomization by randomly selecting split thresholds for

each feature, contrasting with traditional methods that optimize for the best threshold. The performance of random forests, often compared favorably to boosting and arcing algorithms, highlights the effectiveness of variance reduction as a strategy for improving generalization error [8].

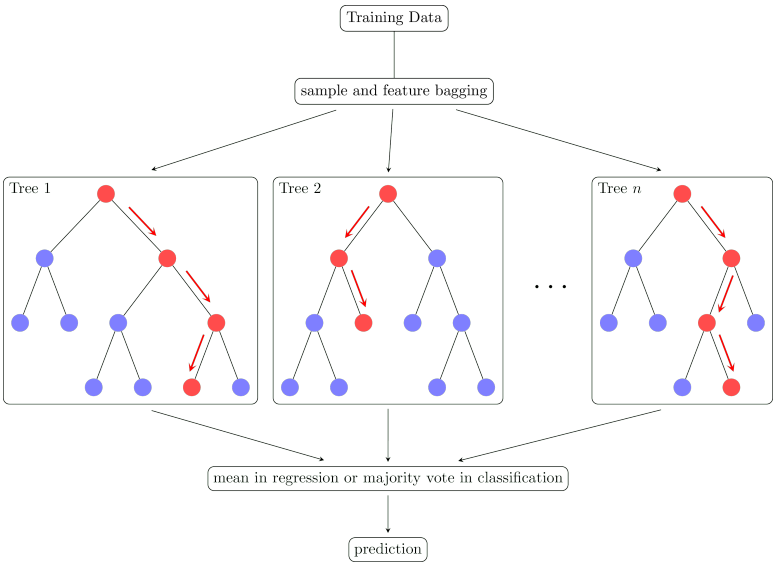


**Figura 3:** Random Forest

### 3.5.2. Stochastic Gradient Descent

Stochastic Gradient Descent (SGD) is a powerful machine learning technique widely used for optimizing classifiers, with the goal of minimizing classification errors. Instead of using the entire training dataset $D_n$ to update the classifier's parameters $P$ at once, SGD processes one sample or a small batch of samples at a time. This iterative approach makes SGD computationally efficient, especially for large datasets, as it updates the parameters more frequently, leading to faster convergence. The process begins by transforming text data into numerical feature vectors $\phi(D)$ that capture the essence of the text. These vectors, in conjunction with the classifier's parameters, form the prediction function $f$ that maps input data to its corresponding class $C_k$. SGD aims to refine the classifier's parameters iteratively by minimizing a loss function, which quantifies the discrepancy between predicted and actual class labels [12]. This iterative refinement enhances the classifier's ability to generalize to unseen data. Research suggests that employing a grid-search approach for hyperparameter tuning can notably improve the performance of SGD in classification tasks. This approach systematically explores a range of hyperparameter values to identify the combination that optimizes the classifier's accuracy and execution time. SGD in the context of specific classifiers such as Support Vector Machine (SVM), Logistic Regression, and Perceptron, highlighting its role as an optimization tool rather than a classifier itself [1, 12, 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 rela-

tionships within data due to their hierarchical structure. As shown in Figure 4, a typical MLP consists of an input layer, which feeds data into the network, one or more hidden layers responsible for non-linear transformations of this data, and an output layer, which generates the final predictions or classifications. Each layer comprises interconnected nodes, referred to as neurons, responsible for processing and transmitting information. The connections between these neurons possess associated weights that are meticulously adjusted during the training process using algorithms like backpropagation to minimize discrepancies between the network's predictions and actual target values. This architecture, particularly the number of hidden layers and the neurons within them, significantly impacts the network's capacity to discern complex patterns and generalize from the data. While the sources acknowledge that pinpointing the optimal architecture for a specific problem often necessitates empirical exploration, they suggest a one-hidden-layer network with 2n+1 neurons, where 'n' represents the number of inputs, as a reasonable starting point. For practical applications, utilizing a conjugate gradient-based MLP is often recommended, particularly in the absence of specialized hardware. Beyond classification and regression, MLPs find utility in diverse domains, including function approximation, forecasting, and even performing multiple discriminant analysis when configured with appropriate linear transfer functions [9, 11].

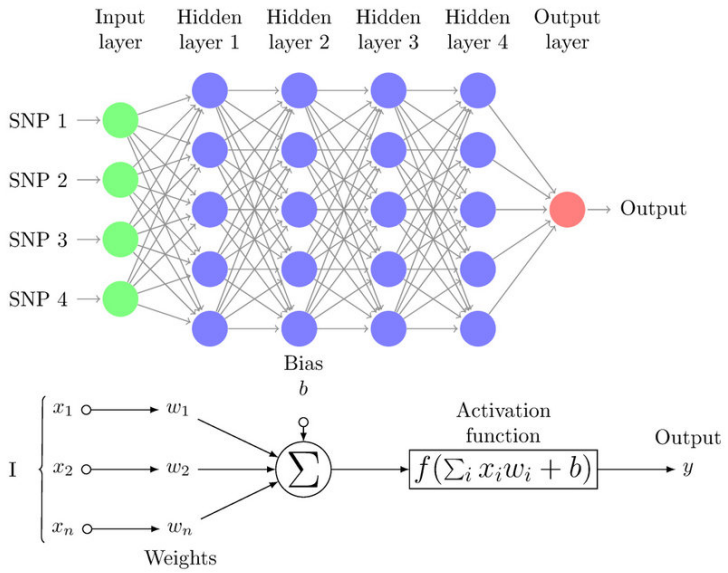


**Figura 4:** Multilayer perceptron

# 4. Implementation and Results

**Cuadro 2:** Label statistics of training set

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

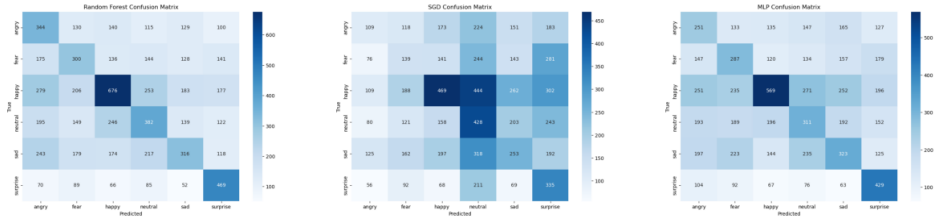


**Figura 5:** Confusion matrix of three model Random Forest, and Stochastic Gradient Descent Classifier (SGDClassifier), Multilayer Perceptron (MLP)

The confusion matrices depicted in Figure 5 offer a nuanced understanding of the strengths and weaknesses of the three emotion recognition models (Random Forest, SGDClassifier, and MLP), mirroring the overall accuracy trends summarized in Table 2. While all models exhibit a moderate ability to recognize 'happy' and 'neutral' expressions, their performance notably declines when distinguishing between more complex emotions such as 'fear,' 'disgust,' and 'surprise.'
The Random Forest model, with an accuracy of 0.35, showcases the most balanced performance across all emotional categories, indicating a relatively robust generalization capability. However, it still struggles with accurately classifying 'fear,' often mistaking it for 'sadness' or 'surprise.' In contrast, the SGDClassifier (accuracy: 0.25) exhibits a more pronounced difficulty in identifying 'sadness' and 'surprise,' frequently mislabeling them as other emotions. This suggests a potential limitation in capturing the subtle features associated with these specific emotional states.
The MLP model, although achieving the highest accuracy for 'happy' expressions (0.31), reveals a tendency to overpredict 'neutral' and demonstrates a significant challenge in recognizing 'fear,' often confusing it with 'anger' or 'sadness.' This points towards a potential oversensitivity to certain facial features or a lack of sufficient representative training data for 'fear.'
Collectively, these findings emphasize the inherent complexity of accurately classifying the full spectrum of human emotions and underscore the need for further research and development in this area. Future efforts could focus on strategies such as incorporating diverse datasets to address class imbalances, exploring alternative feature extraction techniques to capture nuanced emotional cues, and experimenting with ensemble methods that combine the strengths of multiple models to enhance overall performance.

# 5. Conclusion

In this project, we developed a robust system for Facial Emotion Recognition (FER) using a hybrid approach to feature extraction, combining LBP and HOG. By leveraging machine learning models such as MLP, Random Forest, and SGD Classifier, we aimed to accurately classify emotions from facial images. Despite achieving moderate success, the models exhibited varying strengths and weaknesses, particularly in recog-

nizing complex emotions. Future work should focus on addressing class imbalances, exploring alternative feature extraction techniques, and experimenting with ensemble methods to enhance overall performance. This research underscores the potential and challenges of FER, paving the way for further advancements in the field.

# 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 6

[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 6

[4] Joseph Howse, Prateek Joshi, and Michael Beyeler. *Opencv: computer vision projects with python*. Packt Publishing Ltd, 2016. pages 2

[5] 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

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

[7] Shan Li and Weihong Deng. Deep facial expression recognition: A survey. *IEEE transactions on affective computing*, 13(3):1195–1215, 2020. pages 2

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

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

[10] Sakrapee Paisitkriangkrai, Chunhua Shen, and Jian Zhang. Face detection with effective feature extraction. In *Computer Vision–ACCV 2010: 10th Asian Conference on Computer Vision, Queenstown, New Zealand, November 8-12, 2010, Revised Selected Papers, Part III 10*, pages 460–470. Springer, 2011. pages 2

[11] 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 7

[12] Sebastian Ruder. An overview of gradient descent optimization algorithms. *arXiv preprint arXiv:1609.04747*, 2016. pages 6