SD - Facial Landmarks Classification

Project Report

François MULLER & Pierre LAGUE

Repository : https://github.com/Jakcrimson/facial_emotion_classification_sd

SD - Facial Landmarks Classification

Université de Lille

January 2024

Contents

ABSTRACT

In the scope of our project, we will attempt to create a classifier capable of recognizing six facial expressions:happy, fear, surprise, anger, disgust, and sadness. The project aims at developing strong data processing skills and rigorous implementation of supervised learning models with the help of libraries. This project is part of the SD class, taught by Pr. Marc TOMMASI at the University of Lille 1. All of the provided code is directly from the public repository containing the project and is strictly our own.

Keywords Facial Landmark · Classification · Data Engineering · Supervised Learning

1 Introduction

1.1 Objectives

Our primary objective is to predict the emotion column in the emotion.csv file based on the facial points available in the SXXX/omlands.csv files.

To characterize an expression, we can employ various approaches, including:

- Consider the coordinates of the points.
- Consider the movement of the points between the neutral and apex emotion images (the difference between the images).

Faces may appear at different locations in the image, and they may vary in size. It might be interesting to attempt face alignment so that faces are aligned with respect to the eyes and nose (stable points irrespective of the expression). In our experiments, we will use points as they are or place them in a common frame of reference. We will try both approaches.

2 Data Processing

It is important in a supervised learning context to have reliable data on which our algorithms will be based. This is why, we must first proceed to make sure our data is at certain level of quality (for examples, the faces are not centered, the number of instances is different in each classes.

2.1 Face Alignment

When plotting the facial landmarks in a loop we noticed that they were not aligned to the same scale, not centered and slightly off balance for some of them.

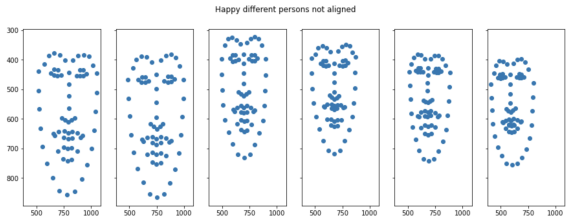

Figure 1: Facial Landmarks not aligned

If our data is not represented within the same coordinate system it will be very difficult for our models to fit in a decent way the data we decided to use to train them. We must arrange

the data in a way that will make it very simple for our models to train on and to generalize from.

In order to do so, we've decided to loop through all of the facial landmarks csv files and automatically center the data in a way where the nose landmarks are centered in 0.

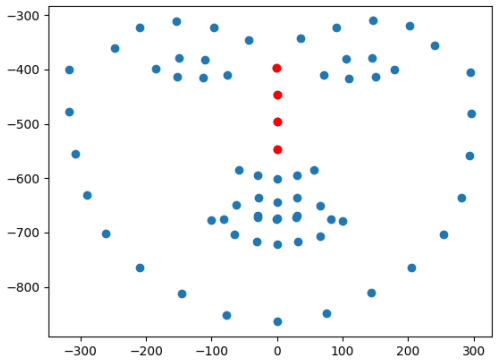

Figure 2: Centered facial landmarks

The algorithm used to center the points is the following :

```
def mean_center_face(face_points: np.ndarray, center_points: np.ndarray =
    np.arange(27, 31), display_face: bool = False) -> np.ndarray:

    center_points_coords = face_points[center_points, :]
    correction = np.array([np.mean(center_points_coords[:, 0]), 0])
    new_face_points = face_points - correction

    return new_face_points
```

Using this function and a loop through the data, we will be able to constitute a clean dataset that has a unique spacial coordinate system, making it's understanding much easier for the model.

2.2 Class Imbalance

When exploring the data and it's associated statistical attributes, we can notice that the number of instances for each class is different, and, rather unequal. We're facing one of the most notorious problem in data engineering and data preprocessing : class imbalance.

Class imbalance is a situation in a classification problem where the classes that are to be predicted are not represented equally in the training dataset. In our context, it means

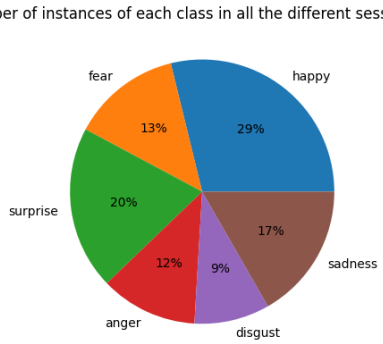

Figure 3: Pie Chart representing the proportion of each class in the dataset

that some facial expression classes (e.g., "fear," "disgust,", "anger" and "sadness") have a significantly lower proportion of samples compared to other classes (e.g., "happy" and "surprise"). This imbalance can be a problem because machine learning models may become biased toward the majority class, leading to poor performance on the minority classes.

There are multiple techniques to address this problem such as :

- Resampling
  - Oversampling : Increase the number of samples in the minority class by duplicating existing samples or generating synthetic samples
  - Undersampling : Decrease the number of samples in the majority class by randomly removing some samples.
- Weighted Loss
  - Adjust the loss function of our machine learning model to give more weight to the minority class. Many machine learning frameworks allow you to assign different weights to classes when training the model (eg. class_weight parameter in classifiers such as RF, SVM, SVC or multiplying the loss by class weights with tensorflow and sparse_categorical_crossentropy ...)
- Stratified Sampling
  - When splitting our data into training and testing sets, we can use stratified sampling to ensure that both sets maintain the same class distribution as the original dataset (parameter stratify=y when using the train_test_split function.)

3 Standard Approach

In this section, we present the implementation of classification methods using the Scikit-Learn Python library, including decision trees, random forests, logistic regression, C-Support Vector Classification, and a multi-layer perceptron regressor. The selection of these classifiers is based on their comprehensive coverage in the Data Science course conducted by Marc TOMMASI, making them well-established choices in the Machine Learning domain, and deemed suitable for our emotion classification problem.

The training and evaluation of the models are conducted on the preprocessed dataset, specifically using facial points that have undergone centering (refer to Section 2.1 on Face Alignment). Only the final faces for each emotion in the saved files are utilized in training and testing, as they exclusively capture the apex of the emotion. This approach contrasts with the initial faces, which may depict emotionless expressions or transitions from neutral to another emotional state. The facial points are persistently stored in alignment with the objectives outlined in Section 1.1.

The dataset is organized into a Pandas DataFrame, encompassing columns for subject, file, emotion (serving as the target variable), and 136 additional columns representing the X and Y coordinates of the 68 facial points. This structured representation facilitates rigorous analysis and model evaluation within a scientific framework.

3.1 Baseline models

The first step is to train and evaluate our models without seeking to optimize their parameters in order to set a baseline (1)

Model	**Cross-Validation Score**	**Score**
Decision Tree	0.565	0.512
Random Forest	0.61	0.605
MLP	0.36	0.319
Logistic Regression	0.824	0.815
SVC	0.369	0.369

Table 1: Performance Scores of Baseline Models

3.2 Grid Search and Optimizing the Models

In this analysis, we present the performance scores derived from a 5-fold cross-validation and the evaluation on unknown data, representing the generalization error for each model.

The training dataset constitutes 66% of the total dataset, with the remaining portion allocated to the test dataset. Cross-validation was performed on the training dataset.

For each model, the cross-validation score surpasses the general error score, with the difference more pronounced for the Decision Tree and MLP models compared to the other classifiers. This discrepancy suggests that the Decision Tree and MLP models may exhibit a tendency to overfit more quickly than the alternative models.

Subsequently, we conducted hyperparameter optimization for the classification models using the grid search method available in scikit-learn. The objective was to identify a set of hyperparameters that enhance model performance within a specified time constraint, as exceeding this limit may yield marginal improvements. The time limit for the grid search was set at 1 hour. The selection of hyperparameters was informed by the scikit-learn documentation, focusing on parameters deemed likely to enhance model efficacy.

- Decision Tree
    1. `criterion` = log_loss
    2. `max_depth` = 13
    3. `max_leaf_nodes` = 100
    4. `min_samples_leaf` = 5
    5. `min_samples_split` = 6
- Random Forest
    1. `criterion` = log_loss
    2. `max_depth` = 9
    3. `max_features` = None
- MLP
    1. `activation` = tanh
    2. `hidden_layer_sizes` = 1000
    3. `max_iter` = 1000
    4. `solver` = lbfgs
- Logistic Regression
    1. `solver` = newton-cg
- SVC
    1. `C` = 0.01
    2. `kernel` = linear

After applyling the results of the grid search in the previous enumeration, we obtain the following results (2).

Model	**Cross-Validation Score**	**Score**
Decision Tree	0.424	0.529
Random Forest	0.58	0.647
MLP	0.743	0.739
Logistic Regression	0.89	0.882
SVC	0.84	0.873

Table 2: Performance Scores of Models with Optimized Hyperparameters

When comparing the results between Table 1 and Table 3, two observations become apparent:

1. Some models, such as Decision Tree, Random Forests, and Logistic Regression, exhibit marginal improvements even with hyperparameter optimization.
2. The Multi-Layer Perceptron (MLP) and Support Vector Classification (SVC) models experience substantial performance boosts, with the MLP achieving an increase of over 40%, and the SVC showing an improvement exceeding 50% after hyperparameter tuning.

Upon analyzing the scores, Logistic Regression emerges as the optimal classification model for the following reasons:

- It attains the highest score on unknown data across all models, both in its basic form and with optimized hyperparameters.
- Similarly, it achieves the highest cross-validation score among all models.
- Logistic Regression exhibits relatively good performance even without hyperparameter tuning, indicating robustness to the dataset characteristics.

However, a notable challenge is the limited size of the dataset, comprising only 357 samples, which may hinder the effective training of certain classifiers. To address this limitation, we implemented a function named `create_data` for dataset augmentation.

3.3 Augmenting the data

The `create_data` function takes the original dataset, the desired final size of the returned dataset (as an integer), and the randomness size (also as an integer) as input arguments.

It computes the number of new facial points needed for each emotion to approximate the target final size, ensuring a balanced representation of emotions in the augmented dataset. Subsequently, the function generates new facial points for each emotion until the desired dataset size is reached, mitigating the class imbalance observed in the initial dataset.

To generate augmented facial points for a given emotion, the process involves randomly selecting an existing face of that emotion from the dataset. For each column representing an X or Y coordinate of a facial point, a random real number is added, chosen from a uniform distribution within the range of [-*randomness size*, *randomness size*]. The magnitude of the randomness size determines the extent of coordinate variation, introducing controlled variability into the newly generated facial points. Once the desired number of new facial points is reached for each emotion, the augmented dataset is returned alongside the original dataset.

An illustrative example (Figure 4) of a newly generated face is provided below, alongside the original face from which it originated:

Generated Face:

$$\begin{bmatrix} X'_1 & Y'_1 \\ X'_2 & Y'_2 \\ \vdots & \vdots \\ X'_n & Y'_n \end{bmatrix}$$

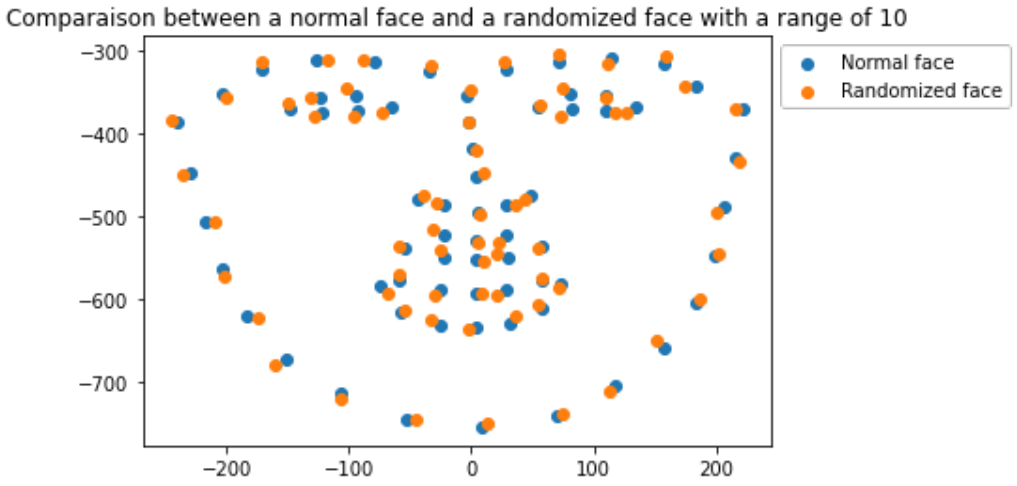

Figure 4: Comparison of a randomized face and a normal face.

To apply this function, the dataset is initially partitioned into training and testing subsets. The new facial points are exclusively generated from faces within the training dataset, ensuring that all facial points in the test dataset are unseen during the training process.

Additionally, no new facial points are derived from faces originating in the test dataset. This approach ensures that the model's performance evaluation on the test dataset remains meaningful, as it has not been exposed to the augmented data during the training phase. Moreover, the separation between training and testing datasets preserves the generalization capability of the model.

Model	**Cross-Validation Score**	**Score**
Decision Tree	0.938	0.508
Random Forest	0.995	0.559
MLP	0.938	0.737
Logistic Regression	0.994	0.855
SVC	0.995	0.838

Table 3: Performance Scores of Models with Augmented Dataset (5000 Faces, Randomness=5)

For the experiments detailed in Table 3, the models were trained using the optimized hyperparameters. Upon comparing the results in Table 2 and Table 3, several observations emerge:

1. The cross-validation score is generally higher for most models. This is expected, as the generated data is a randomized copy of existing data. Therefore, during the validation process, even though the tested data was not part of the training set, similar data may have been utilized.
2. The general score varies among models. The MLP, logistic regression, and decision tree exhibit relatively stable performance, while the random forest and SVC experience a decrease in performance.

These results indicate that all models show signs of overfitting. The high validation score is attributed to the abundance of generated data, which introduces a level of randomization. Despite the overfitting, the general score for each model does not experience a drastic drop, suggesting the potential effectiveness of this method with further refinement.

To optimize the dataset augmentation process, a manual grid search was conducted. The search iterated through various final dataset sizes and randomness sizes. To identify the optimal hyperparameters for the `‘create_data‘` function, the best-performing version of the SVC (previously determined through grid search) was utilized. The cross-validation score and general score were calculated for this SVC model using the augmented training dataset.

The choice of SVC, despite the superiority of logistic regression, was influenced by the significantly longer execution times for cross-validation and scoring with logistic regression. A balance must be struck between the number of generated samples and their randomness. If there are too many identical, insufficiently randomized samples, the risk of overfitting increases. Conversely, if the newly generated facial points are excessively random, they may not adequately represent the underlying emotions, resulting in a lower general score. The goal is to find an optimal balance that ensures an ample amount of suitably randomized data.

3.3.1 Optimized Training Dataset

We found that the hyperparameters for the `‘create_data‘` function for the optimized SVC is 3000 final sample, with a randomness of 12. Here are the results for each optimized models, with the optimized training dataset:

Model	**Cross-Validation Score**	**Score**
Decision Tree	0.769	0.55
Random Forest	0.926	0.703
MLP	0.877	0.864
Logistic Regression	0.867	0.932
SVC	0.912	0.872

Table 4: Performance Scores of Models with Optimized Training Dataset (3000 Faces, Randomness=12)

By analyzing Tables 2, 3 and 4, several key observations can be made:

- The cross-validation score decreased with the optimized dataset, indicating that models no longer overfit the data.
- All scores increased, except for SVC.
- Logistic Regression achieved an impressive score of 0.932, indicating a low error rate of 7%, a significant improvement from the initial threshold of 16

When using bagging, as seen in class, the best logistic regression and SVC models did not yield better results than the individual classifiers.

3.3.2 Confusion Matrices

Confusion matrices provide further insights into the models' performance. We present analyses for the two best models (Logistic Regression and SVC) and the worst-performing model (Decision Tree).

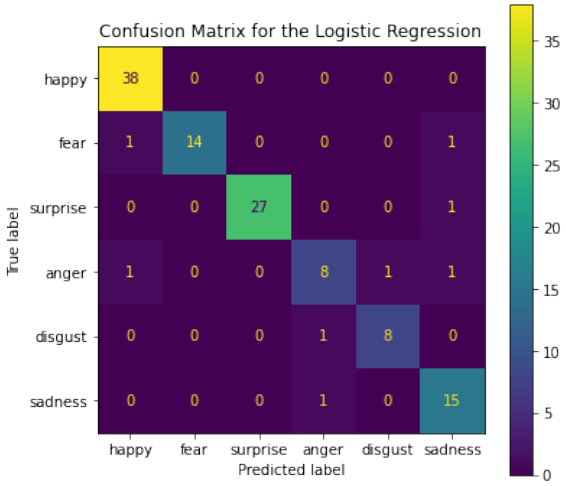

Figure 5: Logistic Regression Confusion Matrix

For the Logistic Regression analysis (Figure 5), it is observed that the model made erroneous predictions for the "anger" class twice, "happy" twice, and "sadness" three times, with one misclassification for "disgust." Notably, the model encounters difficulty in accurately classifying faces expressing anger. Examination of the confusion matrix reveals that the "angry" class exhibits the highest number of misclassifications, totaling three errors. The observed overlap in misclassifications between "disgust," "anger," and "sadness" suggests potential facial point similarities, contributing to the classifier's confusion.

Regarding the Support Vector Classifier (SVC) confusion matrix analysis (Figure 6), a notable observation is that the classifier commits eight errors in classifying facial points as "fear," representing the most significant error count among the predicted values. Additionally, the class with the highest number of errors for the SVC classifier is "sadness," accumulating six errors against ten correct predictions.

The confusion matrix for the Decision Tree, as illustrated in Figure 7, reveals a significant number of errors, providing valuable insights into the model's performance characteristics. Notably, the model demonstrates precision when detecting "surprise" faces, as evidenced by only one false prediction and misclassification of merely two "surprise" faces. However, the class that incurs the most errors is unequivocally "happy," with frequent confusion occurring, particularly with "fear" faces. This observation implies a notable challenge in

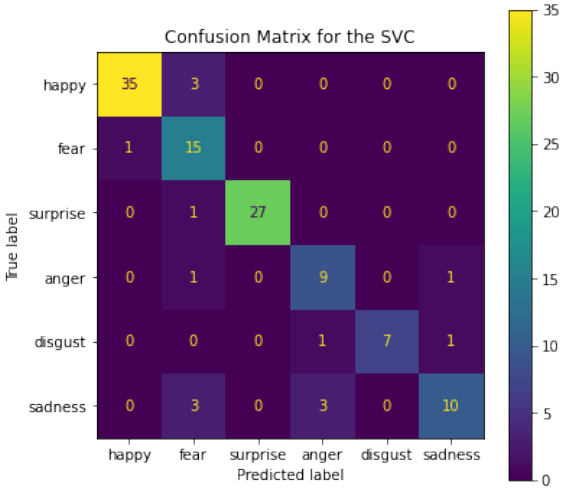

Figure 6: SVC Confusion Matrix

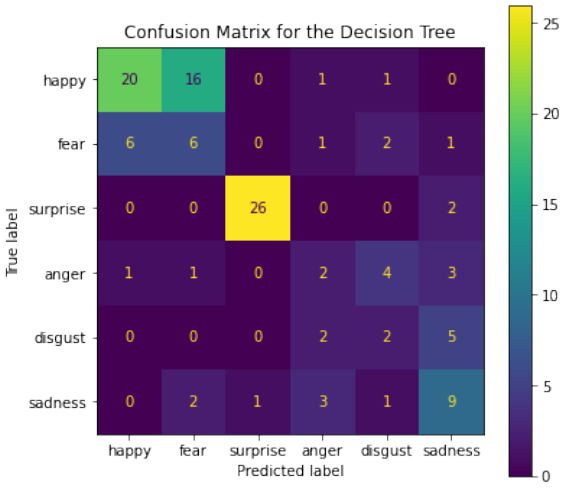

Figure 7: Decision Tree Confusion Matrix

distinguishing between "happy" and "fear" expressions, suggesting a substantial degree of similarity between the facial features associated with these two emotional states.

These insights contribute to a deeper understanding of the models' performance nuances and highlight specific challenges each model faces in correctly classifying certain emotional expressions. Further investigations into feature representations and potential interclass similarities may aid in refining the models for improved accuracy.

4 Deep Learning Approach - CNN

Even if this approach is not specifically required, nor studied during the Data Science class, we thought using Convolutionnal Neural Networks (CNN) would be an elegant way of solving the classification problem. Convolutional Neural Networks (CNNs) are a class of deep neural networks designed for visual perception tasks, leveraging layers of learnable filters to automatically extract and represent spatial features from the data. Characterized by convolutional and pooling layers, CNNs exhibit translation-invariant and parameter-sharing properties, enabling them to efficiently process grid-like structured data, such as images or sequences. The learned hierarchical representations enable effective feature extraction, facilitating tasks like image classification, object detection, and segmentation with remarkable success in various computer vision applications. Given the simple nature of our images, it seemed like a logical solution to explore.

4.1 Data Engineering

Our dataset will be based on images directly extracted from the plotting of the centered facial landmarks coordinates 8. Such image is what will be input into the model.

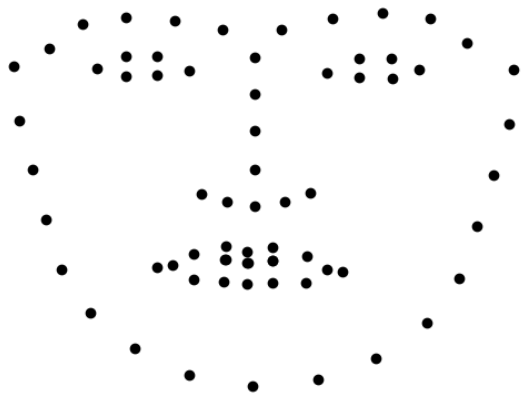

Figure 8: Plotted centered coordinates corresponding to "Anger" in black and white. The data is processed in such way that it is easily interpreted.

All of the images are split into 3 folders, each containing 6 subfolders as such :

```
images
├── test
│   ├── anger
│   ├── disgust
│   ├── fear
│   ├── happy
│   ├── sadness
│   └── surprise
├── train
└── val
```

The data has been divided into 70% for the training set and 15% for both the validation and testing set. We've used the "stratify" parameter in order to maintain class balance within the sets (2.2). This setup allows us to use preprocessing functions from `Keras.preprocessing.Image` to feed our model with images labeled from the folder they are from (`ImageDataGenerator.flow_from_directory`)

4.2 Model Definition

To ensure the viability of our deep learning approach, we emphasize the importance of maintaining a lightweight model with fewer than 4 million trainable parameters and a limited number of convolutional layers. While lightweight pre-trained models like ImageNet or MobileNet offer fine-tuning possibilities, our decision is motivated by the goal to strike a balance between model complexity and efficiency.

Our model architecture, depicted in Figure 9, is designed with 4 Conv2D layers, 4 MaxPooling layers, 2 Dense layers, 1 Flatten layer, and 1 Dropout layer. The total number of trainable parameters is 3,601,478 (13.7 Mb).

The Conv2D layers are employed for feature extraction from the input image, while MaxPooling layers reduce spatial dimensions to mitigate overfitting. The Dropout layer introduces randomness during training, diminishing reliance on specific neurons and thereby preventing overfitting.

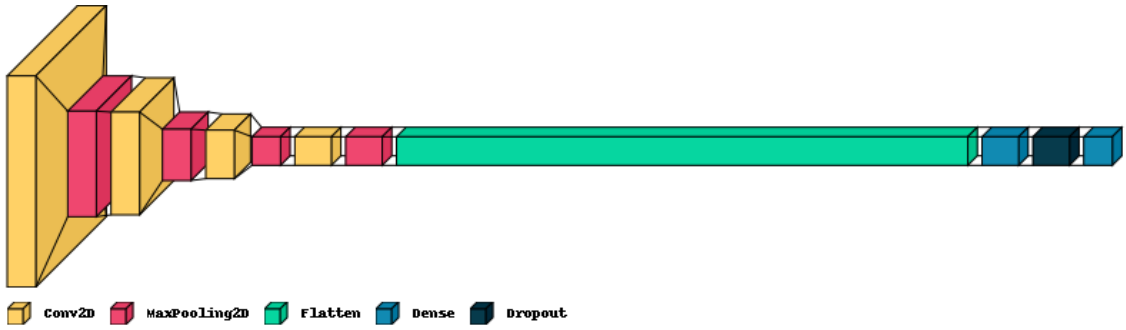

Figure 9: Our model architecture (`visualkeras` lib). 4 Conv2d Layers, 4 MaxPooling Layers, 2 Dense Layers, 1 Flatten Layer and 1 Dropout Layer. Trainable Parameters 3601478 (13.7 Mb)

4.3 Model Tuning

Given the binary nature of our images, we employ Data Augmentation to enhance model robustness and diversify the training set. Experiments, both with and without augmentation, are detailed in Section 4.4. Additionally, we explore an alternative data representation using zoned facial landmarks, as illustrated in Figure 10, providing contextual features for improved data comprehension.

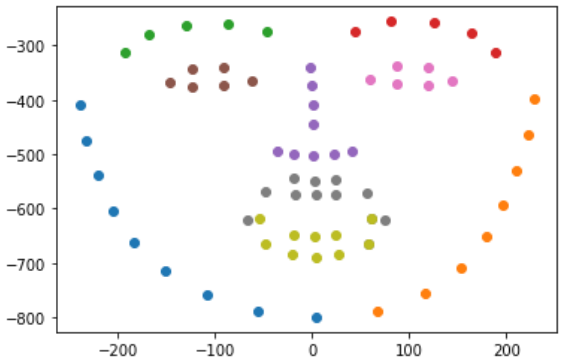

Figure 10: Zoned facial landmarks. Each zone represents a zone in the face. These are contextual features that may help the model understand more the data.

Various callbacks, such as early stopping based on validation accuracy monitoring, are implemented to prevent overfitting during model training. These considerations collectively contribute to the systematic exploration of model tuning strategies in pursuit of optimal performance.

In addition, attempts were made to conduct hyperparameter optimization for the models using a grid search approach. The optimization focused on parameters such as the optimizer,

activation function, and batch size. This was implemented using the KerasClassifier wrapper in conjunction with GridSearchCV. However, it was observed that the computational time required to complete the search was extensive, counted in days. This prolonged duration rendered the pursuit less justifiable, considering the potential marginal gains in accuracy. It is noteworthy that the deep learning model had already demonstrated commendable performance, underscoring the efficacy of the deep learning paradigm as a viable solution, despite its unconventional nature in the context of our course curriculum. This finding reinforces the notion that deep learning represents a potent avenue for solving complex problems, as illustrated by its effectiveness in the context of facial emotion recognition.

4.4 Results

Our experiments encompass various configurations, including those involving data augmentation, colored zones, and different numbers of convolutional layers. The inherent simplicity of our dataset initially led us to anticipate promising results with minimal or no tuning.

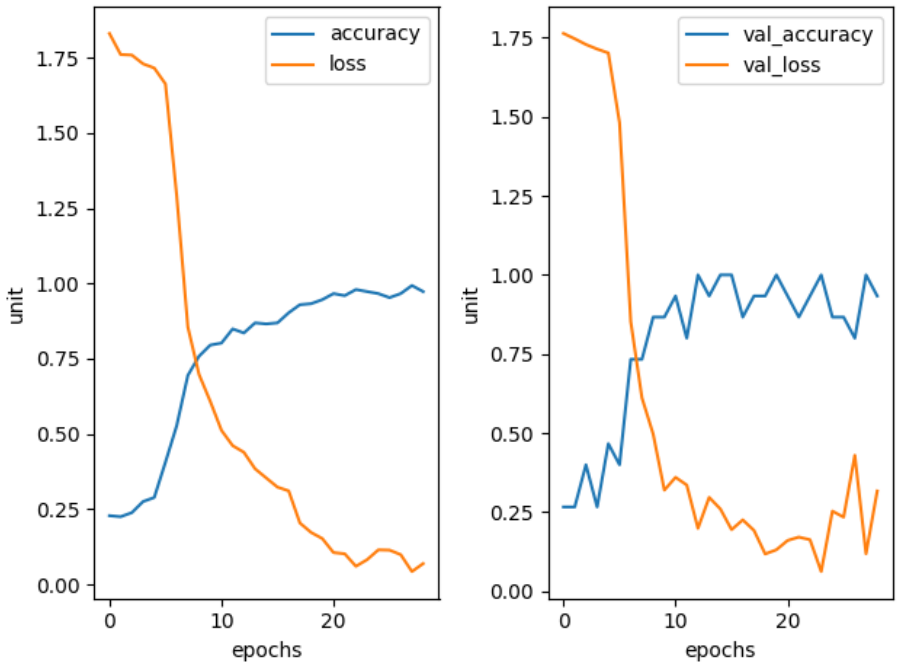

Figure 11: Black and white data, no augmentation. Final validation accuracy of 0.93%. Test accuracy of 0.80%. 30 Epochs. Early stopping at epoch 21 to avoid over-fitting.

Figure 11 illustrates the outcomes of our initial experiment, conducted without any parameter tuning. The employed architecture corresponds to that depicted in Figure 9. However, the achieved validation accuracy plateaued at 0.93%, primarily attributed to the constrained

size of the available dataset. Notably, accuracy stabilizes after the 20th epoch, diverging from the trend observed in the subsequent experiment depicted in Figure 12.

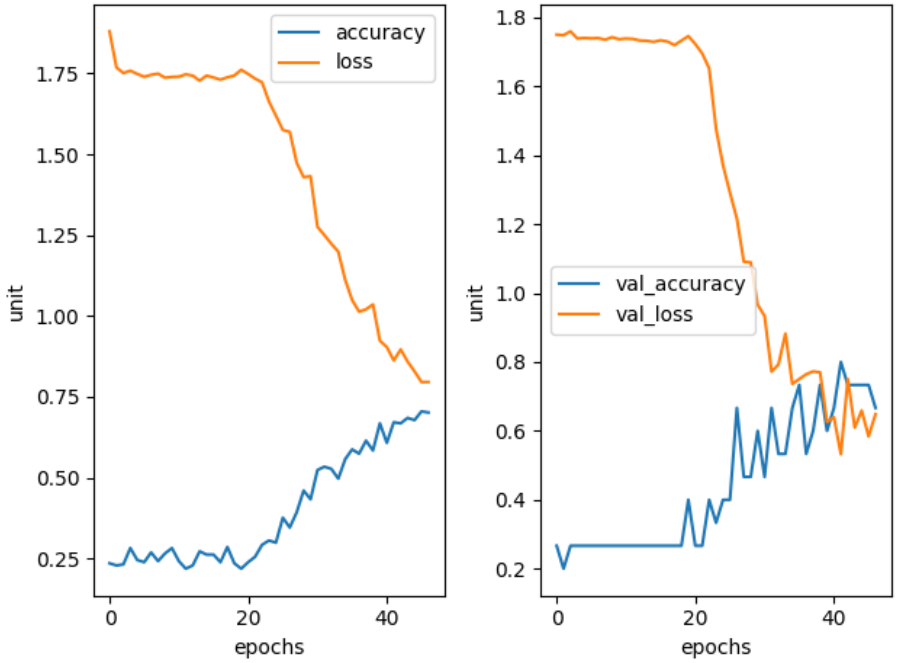

Figure 12: Color zoned data, no augmentation. Final validation accuracy of 0.69%. Test accuracy of 0.60%. Early stopping at epoch 42 to avoid over-fitting.

In the experiment involving colored zones (Figure 12), despite incorporating contextual features (as shown in Figure 10), the model's performance falls short of expectations. The convergence is comparatively slower, reaching a plateau after the 30th epoch. This phenomenon may be attributed to the potential lack of informativeness in the colored zones, hindering the model's ability to interpret them as significant features.

Subsequent experiments revealed that data augmentation prolongs convergence towards significant scores (approximately 70 epochs), compared to the rapid convergence of the untuned model (around 20 epochs). This delay in convergence can be attributed to the increased complexity of the augmented data, which involves rotations, zooms, flips, among other transformations. Despite the extended convergence time, data augmentation results in a more robust model when exposed to poor-quality or noisy data.

The experience depicted in Figure 13 shows that the model is not overfitting, because the training accuracy is lower than the validation accuracy. The evaluation on the test set shows the same result, the model is in fact much more robust and well generalized.

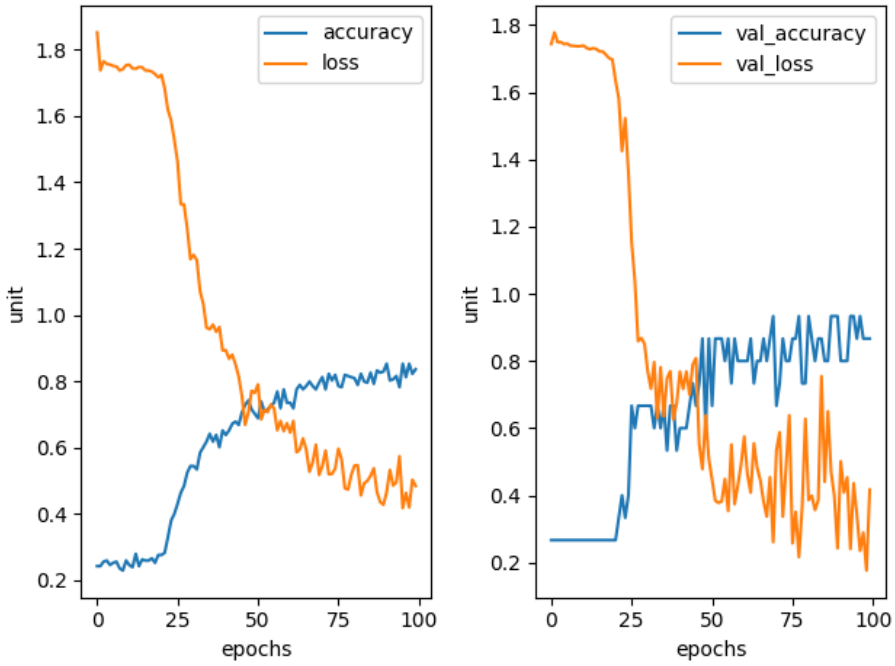

Figure 13: Data Augmentation scores. The model takes longer to converge and reaches a peak validation accuracy of 1.00%, and 0.86% on the test set. No callbacks were employed as the model required extended convergence time.

5 Discussion

The outcomes presented in Section 3.3.2 for standard methods and Section 4.4 for the deep learning approach underscore the efficacy of each strategy in addressing our problem. While both approaches have yielded significant and conclusive results on our test set, there remains room for improvement in each methodology. A recurrent challenge encountered throughout the experiments pertains to the lack of data within our training set. Mitigating this challenge involved employing diverse techniques, including both manual and automatic data augmentation.

Regarding manual augmentation, potential enhancements could involve devising methods to horizontally scale specific facial regions crucial for emotion classification (e.g., mouth, eyes, jaw). Another avenue for improvement in augmentation lies in the introduction of randomness derived from a Gaussian distribution with a defined range (e.g., -12, 12). This would allow for the modification of facial landmarks within a specified range, introducing controlled "noise" to the data. Such an approach has the potential to build a robust model capable of handling noisy or inaccurately sampled data.

Another recurrent challenge pertains to the considerable time invested in optimizing models. While the time invested proved worthwhile for the gain in accuracy achieved by standard methods, the computational demands for deep learning methods are inherently higher due to their complexity. As a consequence, careful consideration must be given to the trade-off between computation time and the resulting accuracy improvements when optimizing deep learning models.

These observations underscore the importance of continuous refinement and exploration of methodologies to address inherent challenges, ultimately contributing to the advancement of emotion recognition models.

6 Conclusion

Throughout this project, a comprehensive deployment of various classification models on our dataset was undertaken. The initial phase involved data engineering to render the dataset comprehensible and amenable to analysis. Subsequently, baseline models were established, serving as unoptimized benchmarks. These baseline models were then subjected to optimization through Grid Search and Cross Validation techniques, resulting in a notable improvement in performance, with accuracy gains of up to 50% observed on the test set. The standard methodologies employed in this optimization process were found to be both explainable and effective in addressing the problem at hand.

Additionally, we explored a Deep Learning approach, particularly convolutional neural networks (CNNs) for direct image processing. Remarkably, this method demonstrated significant performance even with minimal or no hyperparameter tuning. It further facilitated the exploration of novel libraries and techniques to address challenges such as overfitting and data scarcity.

In conclusion, this project has been instructive in guiding us through the process of exploring solutions within the scope of our academic curriculum. It involved the rigorous application of scientific reasoning, from the foundational stages of data preprocessing to the implementation and optimization of diverse classification models.