Sentiment Images Classifier using Agglomerative Clustering and HDBSCAN

1. Dataset

The dataset contains 1347 images with happy and sad people (878 images from happy class and 469 form the sad class). The images are in RGB format and have different sizes. The goal of this project is to use clustering algorithms and try to fit the images in the corresponding clusters, having also the true labels. In order to evaluate the clustering models the random chance score and classification algorithms are used. A classifier needs data to be splitted in train, validation and test so as to train the data only on 0.7 of the entire dataset, tune the model using the validation set 0.15 and then predict on the test set, the rest of 0.15.

The train-test-validation split was performed taking into consideration the imbalanced dataset, the classes' proportionality, the happy class having more labels than the sad one.

2. Preprocessing step

By using tf.keras.utils.image_dataset_from_directory from Tensorflow library, the images are stored in a dataset having all same size, that implied resizing some of them. Also, this function splits the dataset into batches of given size. In order to scale the images, each pixel from the image was split by 255, resulting values in the interval [0,1].

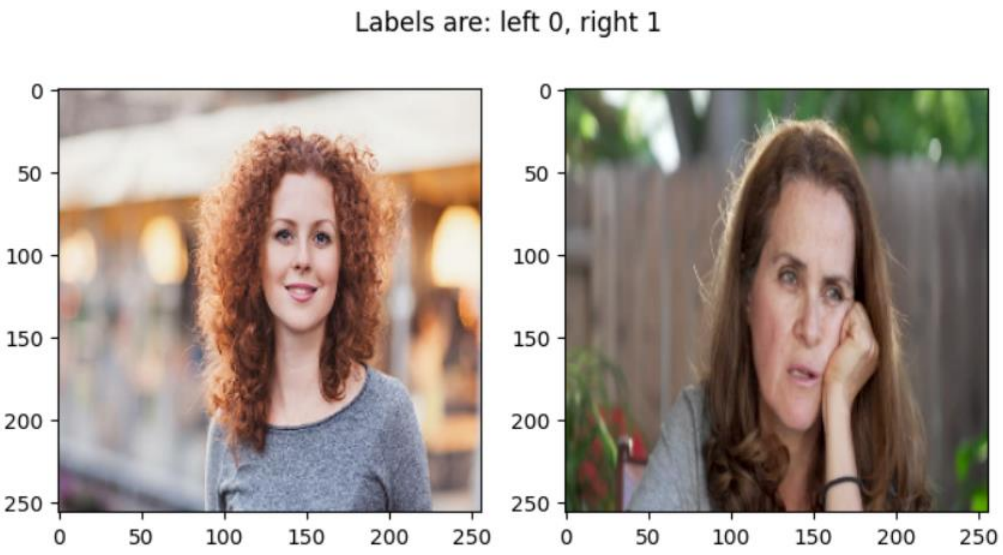

Fig. 1: Representative images of each class (0-happy, 1-sad)

The first feature extraction method was performed using VGG16 pretrained model. The top fully connected layers were excluded, as this model is only used to generate features and not to train on them. By summarizing this model, it has an input layer, Conv2D layers followed by a MaxPooling2D layer. Every channel of every image goes through the convolution operation with a kernel and then the relu activation function is applied which limits the output of a pixel to be 0 or 1. For each image, there are generated 32 new images with each filter. The next layer is the MaxPooling2D which decreases the spatial dimension of the input, by aplying a pooling operation which selects the maximum value of that particular window of size given by the pooling matrix. The last layer added, is a GlobalMaxPooling2D that for each channel of the input feature map, it computes the maximum value across all spatial locations.

The VGG model with its layers is used as features extractor on the dataset. This pre-trained model has already learned hierarchical features from a large dataset (ImageNet) and can be used with the input data chosen.

Second feature extraction technique is Histogram of Oriented Gradients. It extracts the information about edges and their orientation. It computes the image's gradient, the intensity and direction of the pixels then divide it into small cells and compute a histogram for each one. For this part of the project, the library skimage was used to load the images from the folders and preprocess them.

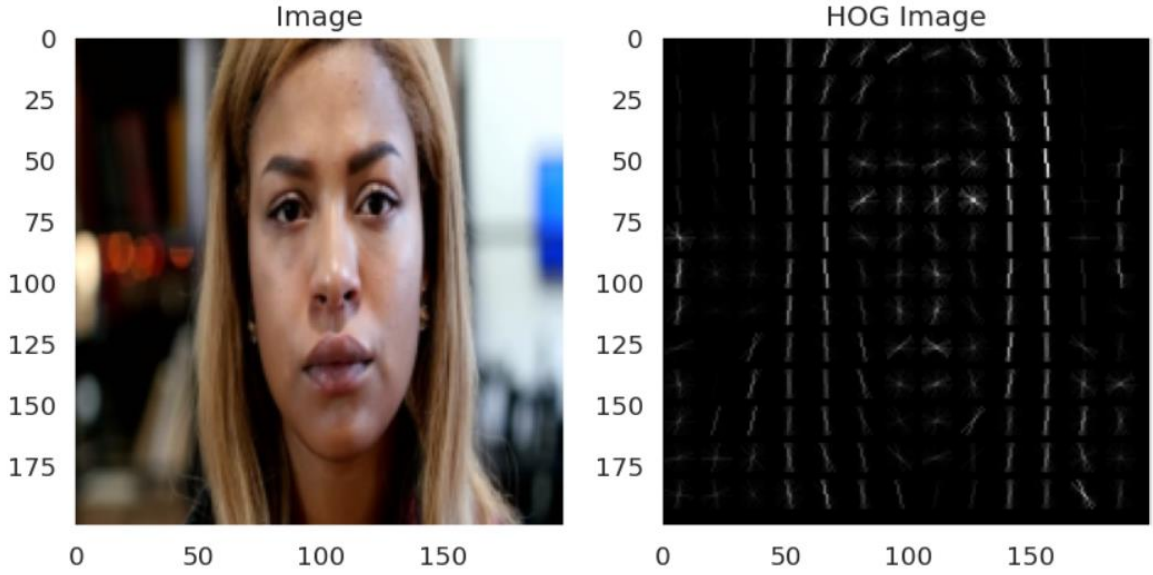

Fig. 2: Image before and after applying HOG

3. Training, Tuning and Testing the ML models

3.1 Supervised Classification Model

For this part, I used Random Forest Classifier that was trained on the training data, tuned using GridSearchCV on the validation data and then tested on unseen data to analyse its performance. The metric used to find the best parametrs is not accuracy, as in an imbalanced dataset this score can be misleading. So I chose, recall which indicates how good is the model at identifying positive instances. After searching for the best parameters, the classification report contains: accuracy 0.59, precision 0.63 for class 0 and 0.25 for class 1, and recall 0.86 for class 0 and 0.08 for class 1. Another approach is to try an oversampling method for the minority class.

In order to address the problem of imbalanced dataset, the method used is Synthetic Minority Oversampling Technique. This works by generating synthetic samples for the minority class. For each minority instance, SMOTE selects k nearest neighbors in the feature space and calculates the differences between the feature values of the instance and its neighbors. It generates synthetic instances by linearly interpolating the feature differences and multiplying them to a random number between 0 and 1 which is added to the original sample.

Visible improvements in the classification score are seen using the RF classification algorithm with SMOTE technique. The classification report contains: accuracy 0.62, precision 0.68 for class 0 and 0.45 for class 1, and recall 0.8 for class 0 and 0.3 for class 1.

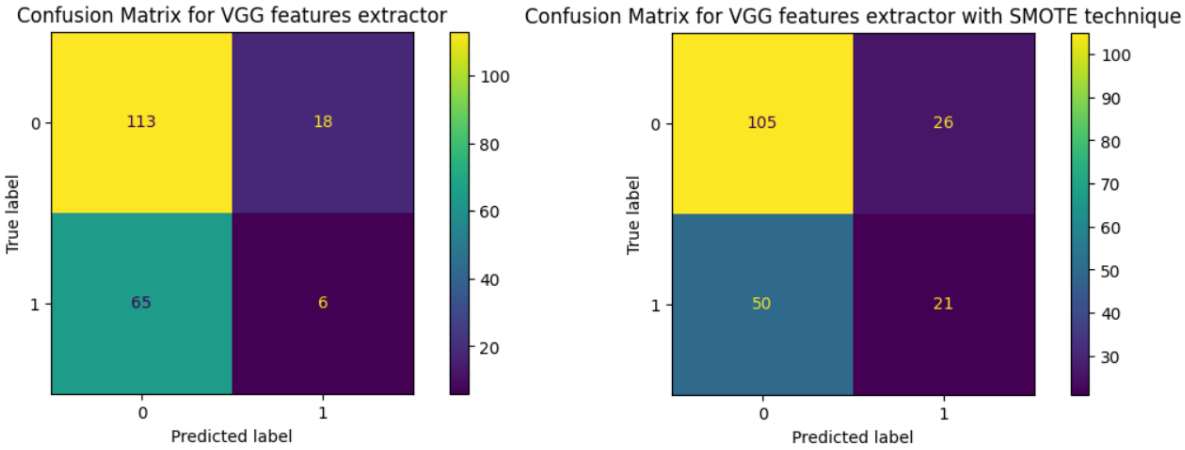

Fig. 3: Random Forest confusion matrix with and without SMOTE

The score obtained from applying Random Forest classification to the HOG features is: accuracy 0.65, precision 0.68 for class 0 and 0.5 for class 1, and recall 0.9 for class 0 and 0.19 for class 1.

Classification report for VGG feature extraction

	precision	recall	f1-score	support
0	0.68	0.90	0.77	130
1	0.50	0.19	0.27	69
accuracy			0.65	199
macro avg	0.59	0.54	0.52	199
weighted avg	0.62	0.65	0.60	199

Classification report for HOG features

	precision	recall	f1-score	support
0.0	0.68	0.80	0.73	131
1.0	0.45	0.30	0.36	71
accuracy			0.62	202
macro avg	0.56	0.55	0.55	202
weighted avg	0.60	0.62	0.60	202

Fig. 4: Random Forest classification report

3.2 Random chance

Baseline models are used as a reference point for comparing the predictions generated by the clustering algorithm. This random chance value should be outperformed by the clustering model's score in order to be considered as an effective model implementation.

Random chance score was performed using a dummy classifier. As the number of classes were imbalanced, the accuracy score is 0.65 and precision 0.65 for class 0 and 0 for class 1, and recall 1 for class 0 and 0 for class 1. By selecting the stratified method which is needed for imbalanced classes, the accuracy is 0.56.

With HOG features, the accuracy was 0.53, precision 0.65 for class 0 and 0.35 for class 1, and recall 0.59 for class 0 and 0.41 for class 1.

3.3 Clustering Models

3.3.1 Agglomerative Clustering

Agglomerative clustering is an hierachical clustering algorithm which starts by treating each sample as a cluster. At each step, most similar clusters are grouped and this procedure is repeated until the desired number of clusters is reached. The hierarchy of clusters is called a dendogram, and it is terminated at a particular height as the number of clusters suggests.

This model has a parameter called n_clusters which in this case can be set to 2, as the dataset has two classes. The train and validation datasests are merged

and are used for the fit_predict part. The result after applying this model on the data is then used as training data for the K Nearest Neighbour algorithm. The unseen test images are then used for prediction.

To validate the clustering models, silhouette score is used. This score is computed using the mean distance between a point and each other data points from the same cluster and the average distance between each point form a cluster and each point from the other clusters. A good model will have the score closer to 1, while a bad model will have closer to -1.

The hyperparameter tuning was performed for the n_clusters by choosing the number of clusters from a range 1 to 10 that has the best silhouette score.

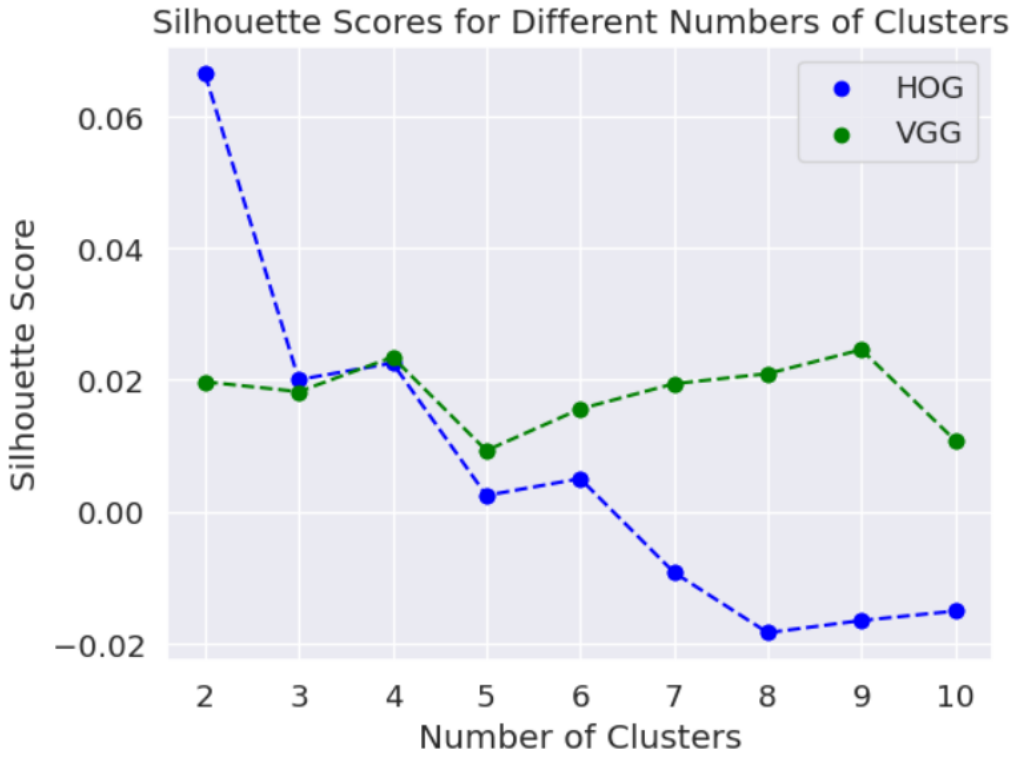

Fig. 5 Silhouette Scores for Different Numbers of Clusters

The scores obtained for the HOG features are: accuracy score 0.64, precision 0.66 for class 0 and 0.41 for class 1, and recall 0.92 for class 0 and 0.1 for class 1. For the VGG petrained model feature extractor, the score obtained is: accuracy score is 0.63 and precision 0.68 for class 0 and 0.46 for class 1, and recall 0.83 for class 0 and 0.27 for class 1. These scores are for n_clusters=2.

VGG feature extraction

	precision	recall	f1-score	support
0.0	0.68	0.83	0.75	131
1.0	0.46	0.27	0.34	71
accuracy			0.63	202
macro avg	0.57	0.55	0.54	202
weighted avg	0.60	0.63	0.60	202

HOG features

	precision	recall	f1-score	support
0	0.66	0.92	0.77	130
1	0.41	0.10	0.16	69
accuracy			0.64	199
macro avg	0.54	0.51	0.47	199
weighted avg	0.57	0.64	0.56	199

Fig. 6 Agglomerative Clustering results

3.3.2 HDBSCAN

HDBSCAN (Hierarchical Denisity Based Spacial Clustering of Applications with Noise) identifies clusters based on the density of data points. It considers regions of higher density as potential clusters. HDBSCAN is robust to noise and can effectively handle outliers. Noise points are often assigned the label -1.

The parameter min_cluster_size was used for hyperparameter tuning. For both feature extraction method, the best silhouette score was obtained for the number 3.

The train and validation datasests are merged and are used for the fit_predict part. The result after applying this model on the data is then used as training data for the K Nearest Neighbour algorithm. The unseen test images are then used for prediction.

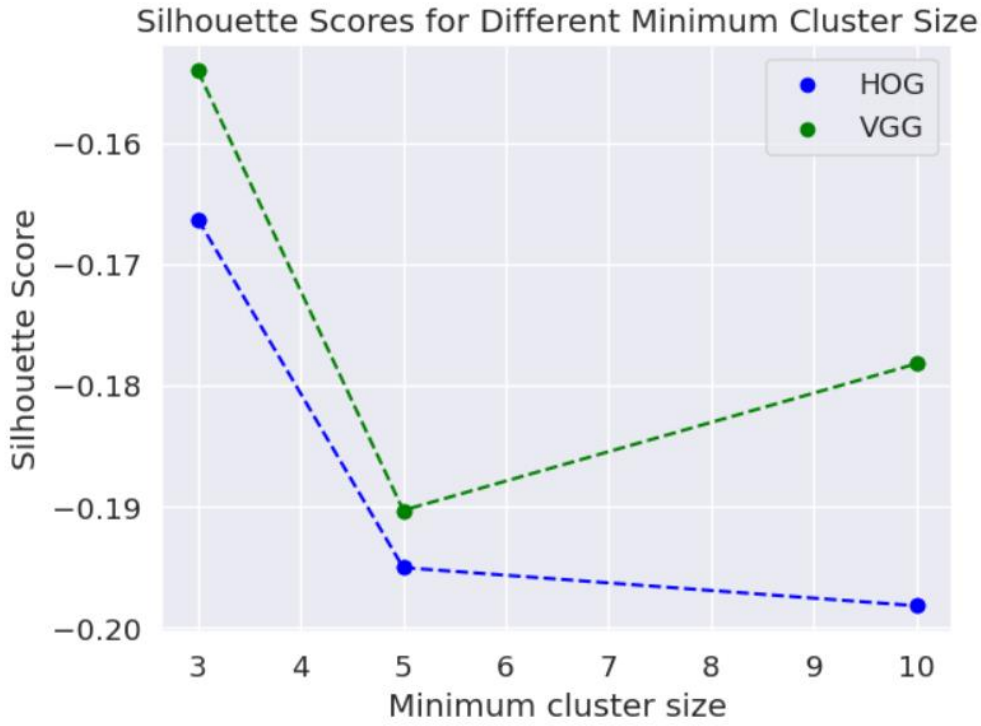

Fig. 7 Silhouette Scores for Different Minimum Cluster Sizes

For this model, the labels are considered noise and only a small number of the training data is detected as part of the happy class and sad class. The possible reason may be that the model there are regions in feature space where the density is low, so HDBSCAN may label those points as noise.

VGG feature extraction

	precision	recall	f1-score	support
0.0	0.65	0.98	0.78	131
1.0	0.00	0.00	0.00	71
accuracy			0.64	202
macro avg	0.32	0.49	0.39	202
weighted avg	0.42	0.64	0.51	202

HOG features

	precision	recall	f1-score	support
-1	0.00	0.00	0.00	0
0	0.61	0.13	0.22	130
1	0.35	0.16	0.22	69
accuracy			0.14	199
macro avg	0.32	0.10	0.15	199
weighted avg	0.52	0.14	0.22	199

Fig. 7 HDBSCAN results

4. Further Improvements

The images do not only contain people's faces, but also other objects in the background which can be eliminated by using algorithms that can detect faces, like Haar Cascade. There are pretrained haar cascade filters that can detect human, parts of human's body, vehicles etc. Once faces are detected, facial expression analysis can be performed. Different facial expressions can provide important clues for sentiment analysis. Eliminating non-human objects from the background can enhance the analysis.