IN3060 COMPUTER VISION: Emotion Recognition

Name: Tanvir H Ahamed, Username: aczj241

Email: tanvir.ahamed@city.ac.uk

Drive link:
https://drive.google.com/drive/folders/1fpUbjZ20LYIQ1rQYXuKBwLdzm7ZLCyf5?usp=sharing

Abstract

This document describes the work performed in developing a computer vision pipeline for emotion recognition. A function called EmotionRecognition was created that can recognize the emotion on the human face of the image passed to it. To achieve this objective, several supervised classifiers were trained with a dataset containing images and their corresponding labels. In total 6 models were trained, in combination of 3 classifiers and 2 feature extractors. The classifiers are: Support Vector Machine (SVM), Multilayer Perceptron (MLP), Random Forest. And the Feature extractors are: Scale-Invariant Feature Transform (SIFT) and Histogram of Oriented Gradients (HOG). This paper explains the theoretical elements of each models, how it was implemented, the results obtained and a comparison of the models.

1 Introduction

Emotion recognition is an important skill of humans, understanding each other's emotions helps to effectively interact with each other. In this generation we interact with machine everyday everywhere so extracting and understanding of emotion has a high importance of the interaction between human and machine(computer)[1]. It has been a subject of research in Computer Vision over a decade, it is a part of image classification task where the images of the faces are the inputs, and the emotion labels are the generated outputs.

The work carried out focused on tackling this emotion recognition task by training testing several image classifiers. The models were first trained on the training dataset folder with their corresponding labels provided in a text file. The model performance was evaluated and method such as Grid Search was used to improve the model performance.

The work was carried out on Google's Colaboratory, Colab in short, it is a Research product that can be used to write and execute any arbitrary python code through the browser and it also provide free access to computing resources including GPU. The Colab files and dataset are available of Google drive. The drive contains 1 zipped dataset file, 3 Colab files, 6 joblib files containing the trained classifiers and 2 test files.

2 Dataset

Classifier model predicts the class a given data belongs to, but first it needs to learn from the given dataset with its labels (supervised learning). So, the first step is to obtain and prepare the dataset to train the model. For emotion recognition, a train and test dataset containing cropped faces with different emotions were provided with their labels in text files. Train folder contains 12271 training

[1] Ieeexplore.ieee.org. 2021. Overview on emotion recognition system. [online] Available at: <https://ieeexplore.ieee.org/document/7292443> [Accessed 4 May 2021].

images and test folder contains 3068 testing images, and the labels were in a text files in the label folder.

Usually, the files are uploaded on the drive and can be accessed from the Colab but when dealing with very large data it can be very slow and cumbersome to access the data from the drive and. So, the whole dataset was first uploaded as a zipped file on the google drive and then it was unzipped directly on the Colab server for faster data handing.

A function called import_dataset was created that takes the images path and label path and imports all the images and add it to a images list and, also imports their labels from the text files available on label folder and add it to the labels list as integer value. This function was used to import the training and testing dataset. Training dataset was used to train the model where the testing dataset was used to evaluate the models' performance. Training dataset set contained 1290 images of surprise, 281 of fear, 717 of disgust, 4772 of happiness, 1982 of sadness, 706 of anger, 2524 of neutral.

Target Class	Label
Surprise	1
Fear	2
Disgust	3
Happiness	4
Sadness	5
Anger	6
Neutral	7

Class table

3 Emotion Recognition

This section provides the details on how different classifier models were trained to tackle the image classification problem of emotion recognition. Each classifier is explained with their theoretical aspects and implementation details. For each implemented methods, qualitative example and quantitative result of performance is discussed.

3.1 Feature extraction

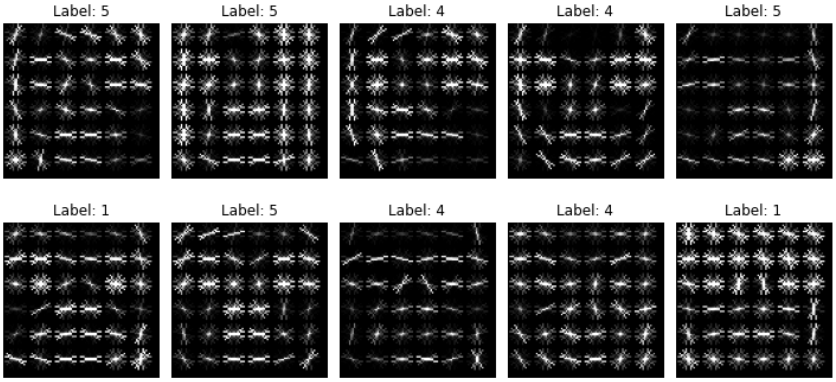
The information of the image contents needs to be provided to the classifier algorithm to train the classifier. These contents, or features, can be colour, texture, shape, position, edges, or corners of the image. It is crucial to choose an ideal feature descriptor algorithm that can extract the features from the images. For emotion recognition HOG and SIFT feature descriptor algorithm were used, then different models were trained with these extracted features. This feature descriptor simplifies the image by extracting useful information and throwing away extraneous information.

3.1.1 HOG

HOG known as Histogram of Oriented Gradient is feature descriptor that focuses on the structure or shape of an object in the image. HOG detects the edges and extracts the gradient and orientation of the edges. The orientations are calculated in localized portions, which means the complete image is broken down into smaller region where the gradient and orientations are calculated for each region.

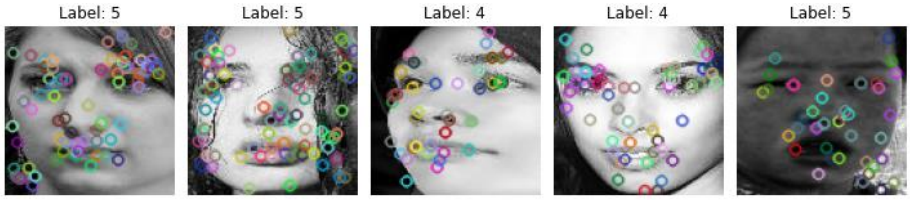
Then histograms of each region are generated separately, hence the name 'Histogram of Oriented Gradient'. [2]

Extract_HOG_featuresDes() function was created that takes the images and corresponding labels and uses the hog library to extract and return the feature descriptors of the images.



3.1.2 SIFT (+bag of features)

SIFT, Scale-Invariant Feature Transform, is future detection algorithm that detects and describes the local features on images in variance to image scale and rotation. Sift.detectandcompute function was used to identify the key points (interest points) in the images and their feature descriptors. SIFT produces gradient vector for each interest points and compute gradient direction histograms.



Then these descriptors were clustered into histogram of code words using Kmeans clustering. In this coursework MiniBatchKmeans were used, since the dataset is very large, and it can divide the dataset into small mini batches to provide faster computation[3].

3.2 Classifiers

The obtained featured descriptors from the above steps with their labels were then passed to classification algorithm to train different models. The models trained were HOG-SVM, HOG-MLP, HOG-Random Forest, SIFT-SVM, SIFT-MLP and SIFT-Random Forest (3 classifier with combination of 2 feature extractors).

---

[2] Descriptor, F., 2021. Feature Descriptor | Hog Descriptor Tutorial. [online] Analytics Vidhya. Available at: <https://www.analyticsvidhya.com/blog/2019/09/feature-engineering-images-introduction-hog-feature-descriptor/#:~:text=HOG%2C%20or%20Histogram%20of%20Oriented,vision%20tasks%20for%20object%20detection.&text=HOG%20is%20able%20to%20provide%20the%20edge%20direction%20as%20well.> [Accessed 7 May 2021].

[3] Medium. 2021. *SIFT (Bag of features) + SVM for classification*. [online] Available at: <https://liverungrow.medium.com/sift-bag-of-features-svm-for-classification-b5f775d8e55f> [Accessed 7 May 2021].

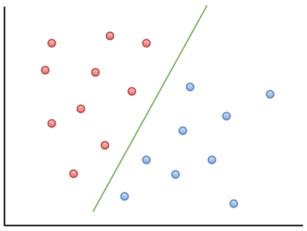
By observing the training dataset can be found that the folder contains consecutive images of same emotion towards the end of the folder. So, the dataset had to be shuffled before passing them to the classifier in order to avoid biases.

First each model was trained with a suitable parameter then evaluated to check the report of precision, recall, f1 and accuracy scores. Then the model was tuned to improve the performance, it was done by Grid search hyper parameter tuning method.

The models trained with HOG feature descriptors are available in this file:
https://colab.research.google.com/drive/1LIxB3FYrLsp_x73xSV4OU_vf_3-bh6FW?usp=sharing

And the models trained with SIFT featured are available in:
https://colab.research.google.com/drive/1ZwEgpJ_ZdBbh6PcYjsFDqk7uUzBBbang?usp=sharing

3.2.1 SVM



SVM, Support Vector Machine, is a supervised machine learning model that uses classification algorithms for two or multiclass problem. After passing the training data to SVM, it is able to learn their feature and identify which data belongs to what class. SVM takes the training data classes and creates a hyperplane that best separate the data. Then any future data that falls to a side of the hyperplane it will be classified as that class. The sklearn.svm.SVC library was used to train the SVM models.

3.2.1.1 HOG-SVM

First a SVM classifier with 'rbf' kernel were defined and it was trained on the extracted HOG features descriptors. It was then predicted on the test dataset and evaluated. The classifier obtained an accuracy of 65%.

	precision	recall	f1-score	support
1	0.64	0.47	0.54	329
2	0.89	0.23	0.37	74
3	0.48	0.06	0.11	160
4	0.73	0.88	0.80	1185
5	0.59	0.49	0.53	478
6	0.64	0.31	0.42	162
7	0.56	0.73	0.64	680
accuracy			0.65	3068
macro avg	0.65	0.45	0.49	3068
weighted avg	0.65	0.65	0.63	3068

Then a grid search was performed which resulted slight improvement in the precision, recall and f1 scores of certain classes but accuracy was the same as 65%. The parameters used for grid search were {'C': [0.1, 1, 10], 'gamma': [1, 0.5, 0.1, 0.01], 'kernel': ['rbf','poly']}

1	0.59	0.50	0.54	329
2	1.00	0.15	0.26	74
3	0.64	0.06	0.10	160
4	0.73	0.89	0.80	1185
5	0.56	0.45	0.50	478
6	0.64	0.33	0.44	162
7	0.57	0.72	0.64	680
accuracy			0.65	3068
macro avg	0.68	0.44	0.47	3068
weighted avg	0.65	0.65	0.62	3068

The qualitative result obtained from HOG-SVM on test dataset:

3.2.1.2 SIFT-SVM

SIFT-SVM was trained the same way as the HOG-SVM but SIFT features descriptors clustered into histograms of code words were passed as the training feature instead. The score obtained from the SIFT-SVM was very low compared to HOG-SVM even after Hyper-parameter tuning. The classifier report shows a downgrade in the scores. Following screen shot shown the report before and after tuning the model:

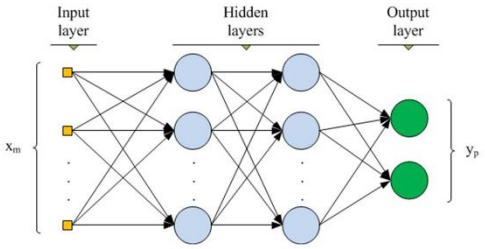
	precision	recall	f1-score	support
1	0.48	0.09	0.15	329
2	0.00	0.00	0.00	74
3	0.00	0.00	0.00	160
4	0.43	0.86	0.57	1184
5	0.39	0.04	0.08	478
6	0.00	0.00	0.00	162
7	0.37	0.32	0.34	679
accuracy			0.42	3066
macro avg	0.24	0.19	0.16	3066
weighted avg	0.36	0.42	0.33	3066

	precision	recall	f1-score	support
1	0.17	0.00	0.01	329
2	0.00	0.00	0.00	74
3	0.00	0.00	0.00	160
4	0.40	0.94	0.56	1184
5	0.00	0.00	0.00	478
6	0.00	0.00	0.00	162
7	0.42	0.17	0.24	679
accuracy			0.40	3066
macro avg	0.14	0.16	0.12	3066
weighted avg	0.27	0.40	0.27	3066

Prediction obtained on the test dataset:

3.2.2 MLP

MLP which stands for Multilayer Perceptron is a class of feedforward artificial neural network (ANN)[4] that maps set of input data onto a set of appropriate outputs. It is consisting of the input layers, hidden layers, and output layers, each layer is fully connected to the following one. The MLP classifier was trained using the sklearn.neural_network.MLPClassifier library.



3.2.2.1 HOG-MLP

HOG MLP was trained with the HOG features Descriptors. First it was trained with a hidden layer size od (50,) with 'sgd' solver. The code was adapted from the lab 6. Evaluating the model on the test dataset an accuracy of 53% was obtained with following scores:

	precision	recall	f1-score	support
1	0.64	0.36	0.46	329
2	0.26	0.23	0.24	74
3	0.23	0.03	0.05	160
4	0.57	0.93	0.71	1185
5	0.50	0.28	0.36	478
6	0.50	0.26	0.34	162
7	0.59	0.46	0.52	680
accuracy			0.56	3068
macro avg	0.47	0.36	0.38	3068
weighted avg	0.54	0.56	0.52	3068

Then a grid search was performed with the hyper parameters of 'hidden_layer_sizes': [(50,50,50), (50,100,50), (100,)], 'activation': ['tanh', 'relu'], 'solver': ['sgd', 'adam'], 'alpha': [0.0001, 0.05], 'learning_rate': ['adaptive']. The performance of the classifier was boosted significantly, {'activation': 'relu', 'alpha': 0.05,

	precision	recall	f1-score	support
1	0.53	0.55	0.54	329
2	0.70	0.31	0.43	74
3	0.38	0.09	0.15	160
4	0.71	0.89	0.79	1185
5	0.52	0.43	0.47	478
6	0.53	0.41	0.46	162
7	0.59	0.55	0.57	680
accuracy			0.63	3068
macro avg	0.56	0.46	0.49	3068
weighted avg	0.61	0.63	0.60	3068

[4] Nair, A., Nair, A. and Nair, A., 2021. A Beginner's Guide To Scikit-Learn's MLPClassifier. [online] Analytics India Magazine. Available at: <https://analyticsindiamag.com/a-beginners-guide-to-scikit-learns-mlpclassifier/> [Accessed 7 May 2021].

'hidden_layer_sizes': (100,), 'learning_rate': 'adaptive', 'solver': 'adam'} being the best Hyper Parameters. The accuracy score was increased to 63%.

An example predicted result obtained from the test dataset was:

3.2.2.2 SIFT-MLP

The SIFT-MLP was trained and tuned using the same parameters as HOG-MLP to provide fair comparison, only SIFT extracted features were used as training dataset. The SIFT-model provided an accuracy of 41% and 42% after tuning, it was completely unable to classify the class 2 and 3 as it was outputting a precision, recall and f1 score of 0.00.

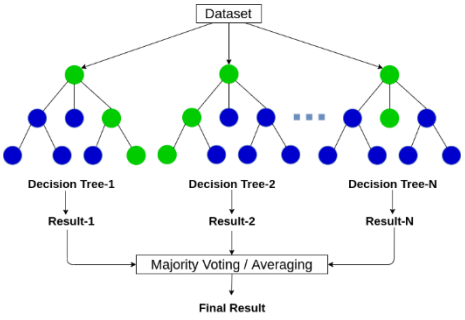
	precision	recall	f1-score	support
1	0.38	0.14	0.20	329
2	0.00	0.00	0.00	74
3	0.00	0.00	0.00	160
4	0.46	0.77	0.58	1184
5	0.29	0.18	0.22	478
6	0.20	0.01	0.01	162
7	0.34	0.33	0.33	679
accuracy			0.41	3066
macro avg	0.24	0.20	0.19	3066
weighted avg	0.35	0.41	0.35	3066

	precision	recall	f1-score	support
1	0.38	0.16	0.22	329
2	0.00	0.00	0.00	74
3	0.00	0.00	0.00	160
4	0.46	0.79	0.58	1184
5	0.30	0.18	0.23	478
6	0.53	0.05	0.09	162
7	0.36	0.30	0.33	679
accuracy			0.42	3066
macro avg	0.29	0.21	0.21	3066
weighted avg	0.37	0.42	0.36	3066

Prediction result on test set:

3.2.3 Random Forest Classifier

Random Forest in supervised machine learning algorithm which can produce great result for classification problems. Random forest classifier creates a set of decision trees from randomly selected subset of training set. It then aggregates the votes from different decision trees to decide the final class of the test object[5].



sklearn.ensemble.RandomForestClassifier were used to create and train the random forest classifier.

3.2.3.1 HOG-Random Forest Classifier

A random forest classifier was created with default parameters values and trained on the HOG feature descriptors. Once tested on the testing set it provided an accuracy score of 56% with the following report:

	precision	recall	f1-score	support
1	0.70	0.22	0.34	329
2	0.00	0.00	0.00	74
3	0.00	0.00	0.00	160
4	0.58	0.92	0.71	1185
5	0.54	0.22	0.32	478
6	0.55	0.10	0.17	162
7	0.50	0.63	0.56	680
accuracy			0.56	3068
macro avg	0.41	0.30	0.30	3068
weighted avg	0.52	0.56	0.49	3068

[5] Medium. 2021. Chapter 5: Random Forest Classifier. [online] Available at: <https://medium.com/machine-learning-101/chapter-5-random-forest-classifier-56dc7425c3e1> [Accessed 7 May 2021].

Grid search was used with the parameters { 'n_estimators': [10,50,100, 150, 250], 'min_samples_split': [2, 4, 6]} to improve the classifer by finding the optimal hyper parameters. It helped to increase the report sore slightly:

	precision	recall	f1-score	support
1	0.71	0.21	0.33	329
2	1.00	0.01	0.03	74
3	0.00	0.00	0.00	160
4	0.59	0.93	0.72	1185
5	0.56	0.22	0.32	478
6	0.50	0.08	0.14	162
7	0.50	0.67	0.57	680
accuracy			0.57	3068
macro avg	0.55	0.30	0.30	3068
weighted avg	0.56	0.57	0.50	3068

An improvement can be seen on the precision of the class 2, it was able to obtain a precision of 100% on the predicted images where the non-tuned classifier was providing 0.00 on precision, recall, f1 scores (was unable to classify).

The result obtained on the test set:

3.2.3.2 SIFT-Random forest Classifier

The random forest was also trained with the SIFT features. It was trained and tuned with the same parameter as HOG-Random forest. But the report result acquired was vey low compared to report acquired with HOG features. The SIFT random forest provided an accuracy score of 40% non-tuned and 41% after hyper parameter tuning, the classifier failed to get any score for 3 classes.

	precision	recall	f1-score	support
1	0.40	0.03	0.06	329
2	0.00	0.00	0.00	74
3	0.00	0.00	0.00	160
4	0.41	0.89	0.57	1184
5	0.27	0.03	0.05	478
6	0.00	0.00	0.00	162
7	0.34	0.22	0.27	679
accuracy			0.40	3066
macro avg	0.20	0.17	0.13	3066
weighted avg	0.32	0.40	0.29	3066

	precision	recall	f1-score	support
1	0.45	0.03	0.06	329
2	0.00	0.00	0.00	74
3	0.00	0.00	0.00	160
4	0.41	0.91	0.57	1184
5	0.39	0.01	0.03	478
6	0.00	0.00	0.00	162
7	0.37	0.23	0.29	679
accuracy			0.41	3066
macro avg	0.23	0.17	0.13	3066
weighted avg	0.35	0.41	0.29	3066

Example of prediction obtained from the test set:

4 EmotionRecognition function

A function called EmptionRecognition was implemented that takes 2 parameters, the syntax being path_to_testset is a string pointing at the test set of the provided dataset and model_type is a string used to load an available trained model. In order to implement this all the trained models were saved on the drive first with the help of joblib library. The function loads the testing dataset from the path provided and chooses 4 random images and extracts their feature descriptors according to the model type. Then it displays the 4 predicted images with its predicted labels and grounds truth (if provided).

This function was used to visualise qualitative results (i.e., predictions) obtained with the different models on the test set images and personal test images (inTheWild).

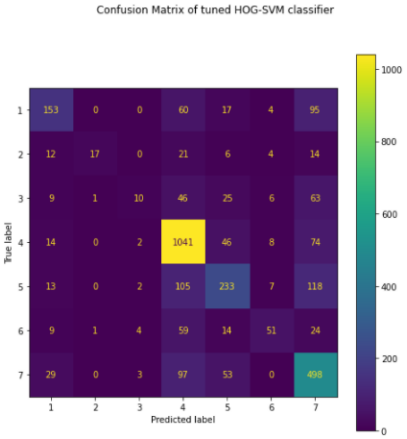
Link : https://colab.research.google.com/drive/1CGLdHvzoiWtIL_fDYQ1ZaQ7yrLP00gro?usp=sharing

Example of prediction on personal dataset:

5 Discussion

Different evaluation technique such as precision, recall, f1 score and confusion matrix was considered to evaluate the models. Precision indicates the number of positive class predictions that belong to the positive class where recall indicates the number of positive class predictions made out of all positive examples in the dataset, and f-Measure provides a single score that balances both the concerns of precision and recall in one number[6]. Confusion matrix is a table that helps to visualize the prediction result of a model, the number of correct and wrong predictions are displayed with count values broken down by each class. All the confusion matrices of the models are available on Colab files.

By observing the result obtained from the test set and comparing to their ground truth, it can be noticed that the models were able to predict at least 2 out of 4 images. The models trained with HOG features were able to perform better than the SIFT features descriptors. The reason being that SIFT had to cluster the features into a histogram of codewords for each image and since all the images were faces it was struggling to cluster the image parts into different classes.



Considering emotion recognition being one of the toughest image classification problem, and out of all the models, HOG-SVM was able to provide a higher score hence it can be considered the best model. But the model is not perfect either, the confusion matrix shows that the model was able predict the class 4 correctly with ease most of the time and the reason can be that the model was imbalanced, it was training on higher number of class 4, so it was leaning to just predict 4 most of the time. Maybe a balanced dataset could have provided a more unbiased model.

6 Conclusion

The project was very challenging and was fun to be able to solve it. I was able to complete 6 models that can predict emotion of a given image of face. I am now very confident to be able to train different machine leaning models on different dataset. As mentioned in the discussion section that the dataset was imbalanced, the performance of the models could be improved by implementing a data balancing technique.

[6] Brownlee, J., 2021. How to Calculate Precision, Recall, and F-Measure for Imbalanced Classification. [online] Machine Learning Mastery. Available at: <https://machinelearningmastery.com/precision-recall-and-f-measure-for-imbalanced-classification/#:~:text=Precision%20quantifies%20the%20number%20of,and%20recall%20in%20one%20number.> [Accessed 7 May 2021].