Face Recognition and Identification
Course Project

Om Kumar (B22CS081) Sumeet S. Patil (B22CS052) Aditya Rathor (B22AI044)

Avani Rai (B22CS094) Swaksh Patwari (B22AI065) Manya (B22CS032)

Abstract

In this project, we explore the effectiveness of feature extraction techniques—Histogram of Oriented Gradients (HOG), Convolutional Neural Networks (CNN), and Local Binary Patterns (LBP)—for improving facial recognition accuracy. We employ these techniques to extract discriminative features from facial images, capturing patterns crucial for accurate identification. Following feature extraction, we evaluate the performance of various classifiers and also optimised hyperparameters wherever required to classify facial images based on their labels(names associated to the images).

The Report describes the performance of traditional Machine Learning techniques in simulating the **Face Recognition unit of brain**. It includes the Workflow of training models,Approaches taken, Results and Inferences.

Contents

I. Introduction

Face Recognition, subset of Bio metric technology, has attracted the Machine Learning scientists from past 50 to 60 years, for its wide range of applications, from User authentication to Attendance Management. Currently, powerful Neural Networks have dominated the field with very high accuracy and reliability. This report however, does not delve into the realm of neural networks. We stick to the **old**, but the **pioneer** Machine Learning techniques.

II. Objective

The objective of this project is to optimize different Machine Learning techniques for Face Recognition. Also, compare their performances' and draw Inference from these results.

III. Background

The dataset used to train all the models is: LFW Dataset

Facial Recognition involves:

- **Face Detection**: Locate and identify the presence and location of faces in the input data.
- **Face Alignment**: Normalize the face's position, orientation, and size to ensure consistent positioning of facial features.
- **Feature Extraction**: Capture and represent the unique characteristics or features of a face, such as facial landmarks.
- **Recognition and Identification**: The model identifies the individual and labels the image according to the dataset it was trained on.

Here, we will be focusing on the latter two tasks, namely, Feature Extraction and Recognition.

IV. Feature Extraction

- HOG is a feature descriptor that captures the distribution of gradients in an image. We resize each image to a fixed size(128*64) and then compute the HOG features using the hog function from the scikit-image library.
- Since the number of features were 3780 so decided to reduce the dimension using After computing HOG features for each image, we apply Principal Component Analysis (PCA) to reduce the dimensionality while retaining 95
- On applying on the training data I was able to get 944
- CNN: We utilize a pre-trained ResNet-50 model to extract deep features from images. The model is loaded and set to evaluation mode, and the last fully connected layer is removed.
- Local Binary Pattern (LBP) is a texture descriptor used for texture classification in images. It provides a robust representation of texture by encoding local patterns of pixel intensities.
- Whereas in case of CNN image cant be constructed as CNN features represent abstract representations of the input image, which are not directly interpretable as pixels.

V. Models

A Decision Tree Classifier

A.1 Approaches Tried

After extracting features, we conducted an analysis of four different scenarios to explore the impact of feature extraction techniques on model performance, which were:

- Taking features of each category separately and training.
- Concatenating features and then training.

For each scenario, we evaluated the performance of the trained models using standard evaluation metrics such as accuracy, precision, recall, and F1-score.

Additionally, we conducted a comparative analysis between each approach to assess the strengths and weaknesses of each approach in terms of classification performance and computational efficiency.

So we divided the data into train and test sets and trained the model, obtaining the following results with Decision Tree:

- Decision tree == accu 0.0034, 40m training time (for LBP)
- Decision tree == accu 0.041, 72m training time (for CNN)
- Decision tree == accu 0.0071, 63m training time (for HOG)

The reason for low accuracy and intense computation was that as there are 13233 images and 5749 people in which there are instances where lot of labels (around 5000) have just 1 image and after number of images for a person exceeded 10 a increasing pattern in number of images with number of individual around 1.

add people vs image

Here we faced 2 problems :

- that people with images less than 10 are of no use for training and testing as unessecary decision boundaries created causing problem for others images in prediction and high possiblity of a image to occur in only the testing set and not in the training set that will always be classified as wrong
- Next we are having a problem that people with just 10 images v/s people with image greater then 200 so showing a class imbalance type of scenario.

Hence to solve the problem of 1st we considered a limit of min images to have for a label to be considered in the dataset else ignore For the 2nd problem we created 2 pathways:

- While filtering dataset we took labels which had images greater than 50 and in
- second pathway we had labels which had images between 10-50

Experimental Setup:

Now for labels with images greater than 50:

Subsequently, the dataset was divided into training and test sets using an 80:20 ratio. The training data was then normalized.

Upon examining the shape of the training data (`X_train`), it was observed to be (1248, 3248).

Such dimensions indicated that training a decision tree model with the best parameter is difficult. Even though you will able to apply grid search CV to find the best parameter, you won't have an idea on the best domain to choose for the parameter.

For instance, during training, I found the decision tree max height with other values as default can go up to 22 and max leaf node can go up to 1249 starting from 1 till max samples in it.

Determining Hyperparameters:

To address the complexity challenge, a systematic approach was adopted to determine the optimal hyperparameters for the decision tree model:

- **Max Depth:**
  - Initially, the max depth parameter was determined using default values for the decision tree on the training sample, as the default is None and goes until it reaches one leaf node or min sample split fails, which has a default value of 2.
  - This max depth value was then used as a reference to iteratively train the model with varying max depth values, incremented in intervals of 2.
  - Prior to training, the data was split into training and validation subsets to evaluate the model's accuracy on unseen data. This iterative process provided insights into the optimal max depth range for the decision tree, which was subsequently used as a reference range for hyperparameter tuning in the GridSearchCV.
- **Max Leaf Nodes, Min Sample Split, and Min Sample Leaf:**

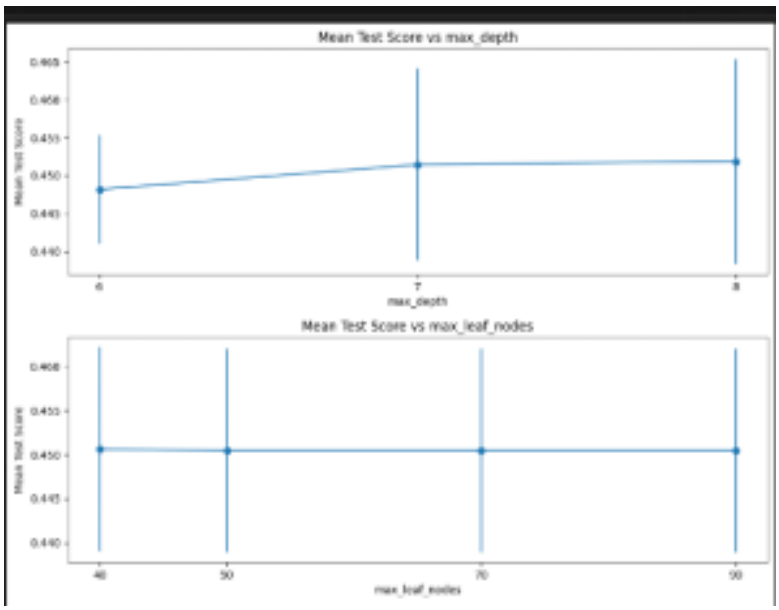


Figure 1: Confusion Matrix with Random Forest

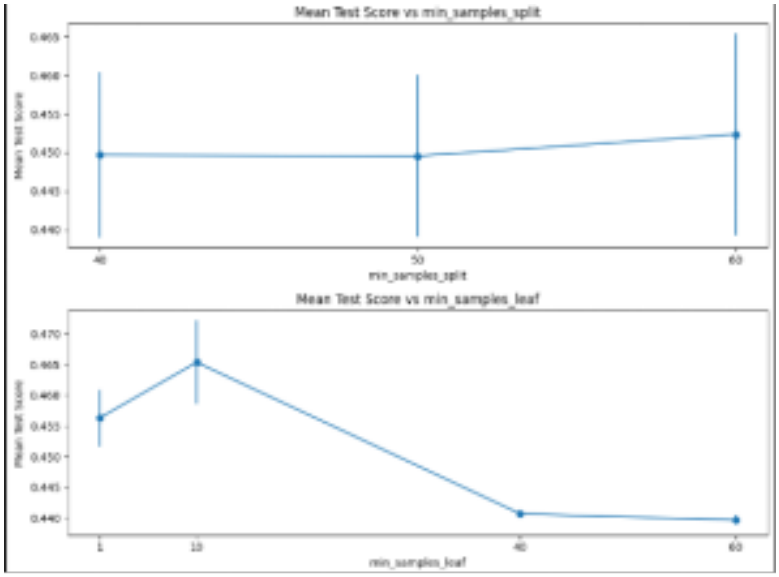


Figure 2: Confusion Matrix with Random Forest

- Similar to the max depth, the hyperparameters max leaf nodes, min samples split, and min samples leaf were explored.
- Starting from their minimum possible values (1 for min samples split, and varying intervals for the others) up to the maximum number of samples, an iterative approach was employed to gain a preliminary understanding of their optimal values.

Hyperparameter Tuning: Utilizing the identified hyperparameter ranges, GridSearchCV was employed to perform exhaustive hyperparameter tuning with cross-validation (cv=5) and accuracy as the evaluation metric.

Individual Feature Analysis:

While the previous analysis focused on concatenated features, it was imperative to investigate how the model performs when trained on individual feature sets such as HOG, CNN, and LBP. To achieve this, the dataset was partitioned based on the indices corresponding to each feature category.

B Random Forest

Random Forest is a type of Ensemble Learning that is composed of Decision Trees. It uses boot strapping or bagging, in short to train the model. Bagging is a type of data split, wherein data is split into small batches

Features	Precision	Recall	f1-Score	Accuracy
CNN	0.19	0.35	0.23	0.35
HoG	0.14	0.32	0.2	0.32
LBP	0.13	0.31	0.18	0.31
ALL	0.4	0.4	0.4	0.4

Table 1: Accuracy of Best Trained Decision Tree

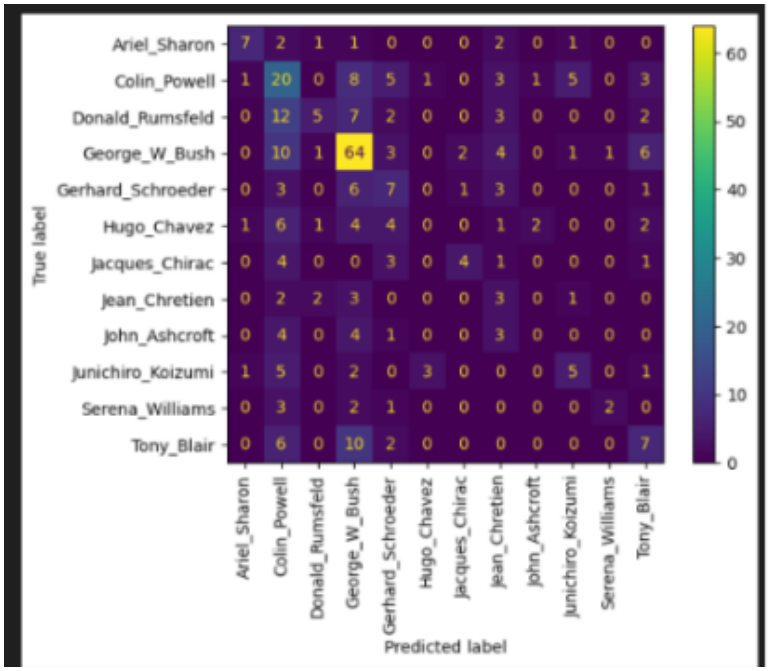


Figure 3: Confusion Matrix with Random Forest

and each decision tree here is trained independently on a single batch. Later, while predicting, all the trees give their predictions. Predictions are aggregated to give the final prediction.

B.1 Approach Used to train models

We had trained the Random Forest on all the three features independently. This was done to see the features that actually are important in this scenario. Later, we concatenated all the features together. They were normalized and finally the model was trained on these features.

Looking into the magnitude of data, with 10,586 records and 3248 features, it was impractical to train the model. Thus, we chose two such minimum number of images criteria, 50 and 70. It was found that 12 classes had images above 50 and around 7 classes above 70.

This change helped in exponentially reducing the runtime of the model, and also improved accuracy. Here we present numbers for 50 images criteria, while the code contains outcomes of 70 images criteria as well. We assume 50 images criteria to be a more general one than 70 images criteria, dealing with only 7 classes. This made us put up 50 images criteria into the report.

When trained on all records, the link to the train is : Pre Training

B.2 Hyper Parameter tuning

A Random Forest has following important hyper parameters

- Maximum depth of each tree
- Maximum number of Leaf Nodes
- Minimum number of Leaf Nodes
- Number of Decision Trees

All these hyper parameters do not have a fixed range.Thus,Hyper parameter tuning was done as a two step process. Firstly, we found the best value of the hyper parameter independent of the other hyper parameters. Later, considering relevant margin for each hyper parameter, Grid Search was applied to find the best permutation of the hyper parameters.

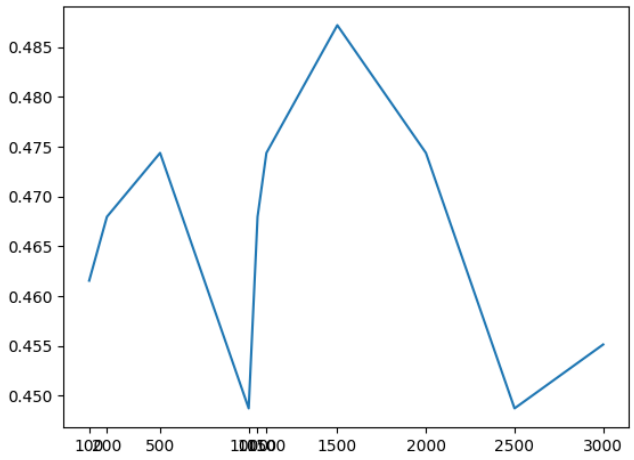
B.3 Outcomes

Following were the outcomes of training:

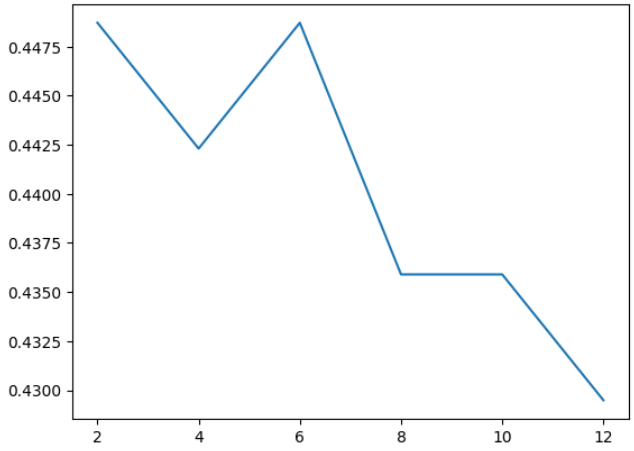
- The best parameters found out are : 'max_depth': 2450, 'max_leaf_nodes': 500, 'min_samples_leaf': 2, 'n_estimators': 50
- The Best Cross Validation score: 0.4917682543672238
- Features when individually taken, CNN extracted features has the best performance of about **48 %**
- Remaining two features have an accuracy of about 35 % each.
- Graphs of Accuracy v/s Hyper parameters can be seen below
- Table 2 shows the performance metrics of the best trained model using all features after hyper parameter tuning.

C Random Landscape

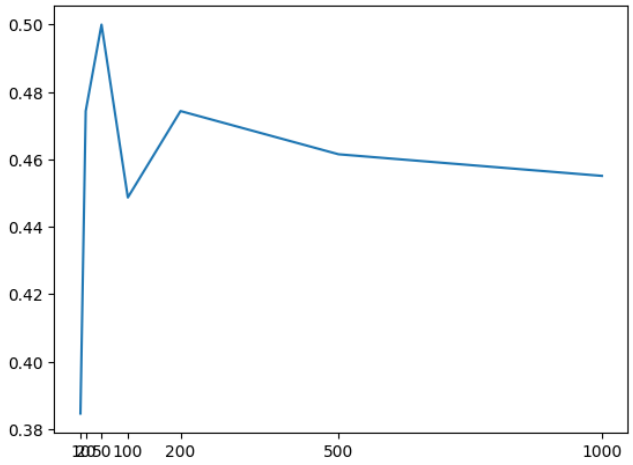
A new name of old kind of Ensemble Learning. We have created an ensemble of Random Forests in this model. This class has been written from scratch. We created this class keeping in mind that it could be possible to get good performance when we combine many random forests. However, it was realised that this could not happen.



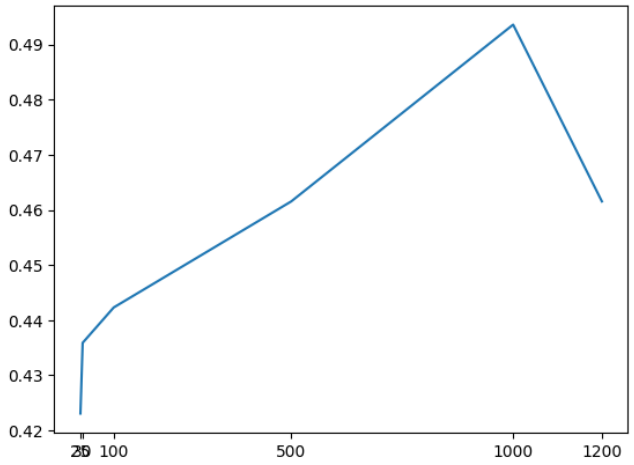
(a) Accuracy v/s Max Depth



(b) Accuracy v/s Minimum Number of Leaves



(a) Accuracy v/s Number of Decision Trees



(b) Accuracy v/s Maximum number of Leaf Nodes

Features	**Precision**	**Recall**	**f1-Score**	**Accuracy**
CNN	0.48	0.47	0.36	0.47
HoG	0.19	0.35	0.21	0.35
LBP	0.19	0.34	0.22	0.34
ALL	0.43	0.42	0.32	0.42

Table 2: Accuracy of Best Trained Random Forest

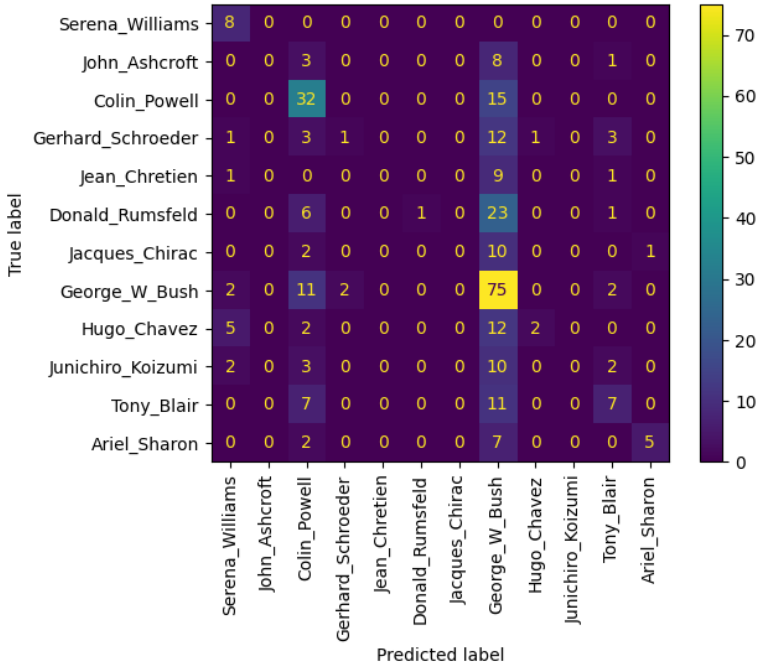


Figure 4: Confusion Matrix with Random Forest

C.1 Approach Used to train models

We have allowed two types of training here. One, all the random forests are trained on all the records and features independently.
Another, taking a Bagging Approach, and training every Random Forest on a particular batch. In bagging,we are splitting it vertically, meaning that every Random forest is trained on different features.

Reason behind this split is that we have seen previously that George W Bush has the maximum number of records, so if we split in record wise, each forest would end up getting more concentrated batch of that class. Thus, each forest will end up getting aligned to George W Bush.

C.2 Hyper Parameter tuning

A Random Landscape has following important hyper parameters

- Number of Random Forests
- Bagging

C.3 Outcomes

Following were the outcomes of training:

- Features when individually taken, CNN extracted features has the best performance of about **38 %**
- After tuning the Hyper Parameters the best model with validation accuracy of about 48 % was obtained.
- Table 3 shows the performance metrics of the best trained model using all features after hyper parameter tuning.

D Artificial Neural Network(ANN)

ANN is type of network inspired by the biological Neural Network Of human brains. It uses different layers of neurons to train the model. These neurons are densely connected to their adjacent layers.As we have extracted 3 different features so we will be training our neural network on all these 3 features separately and also 1 with all these features concatenated.

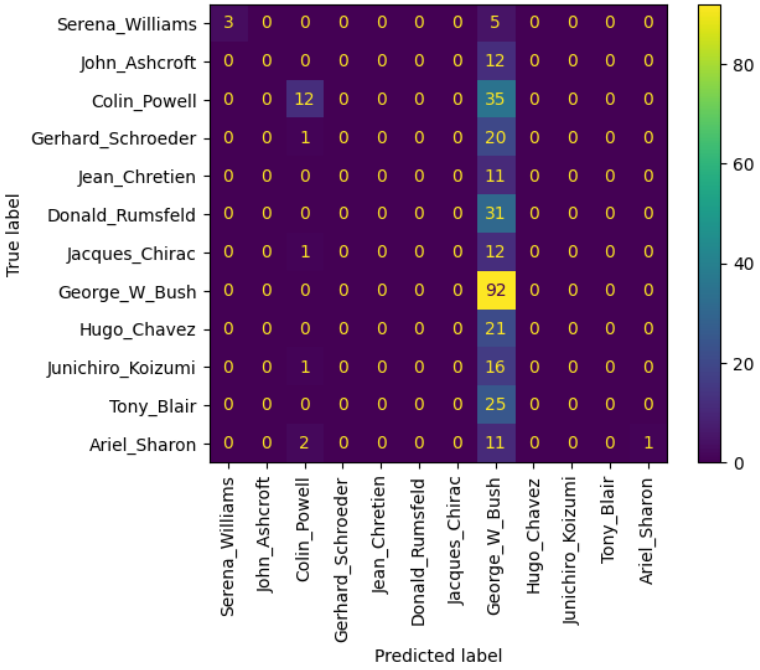


Figure 5: Confusion Matrix with Random Landscape

Features	Precision	Recall	f1-Score	Accuracy
CNN	0.37	0.38	0.26	0.38
HoG	0.09	0.29	0.13	0.29
LBP	0.22	0.31	0.18	0.31
ALL	0.09	0.29	0.13	0.29

Table 3: Accuracy of Best Trained Random Landscapes

D.1 Approach Used to train models

For the building of ANN model we have created a sequential **linear stack of layers**. So we built model that consists of layers with different dense neurons and different activation functions. Now the 1st layer will be the input and the last is being the output one and the between one being hidden layers which helping to train the network efficiently.
At the end the main best model we get have sequence of layers with the decreasing order of the neurons so we can say that we have a Converging Neural Network. **Dropout** we add in between of the layers that helps to prevent the over-fitting of the neural networks.
Also using these different activation functions causes non linearity in our network. On our output layer we have used **softmax** activation function which majorly used for the multi class classification as the number of unique classes in the dataset is more than 2 so we need to do the multi-class classification.
This **softmax** function will give a series(based on unique classes) of outputs for each sample of data giving the output probabilities scores of each unique classes(size = no of unique classes). So the class with max probability we become the predicted label.
For optimizing the Neural Network we used the Adam optimizer. We have chosen sparse categorical cross entropy as our loss function, which is suitable for classification tasks with integer-encoded class labels. As our target is also integer encoded.

Also here as we are considering the only the classes which have count ¿=50 so this models are handling only 12 classes which might be very less.So we have separately trained the model on n of classes between 10 to 50 so that our model can have more classes i.e. 146. But here accuracy we get is **33%** only. So it depends on decision whether to get more classes and less accuracy or less classes with high accuracy.

D.2 Hyper Parameter tuning

In ANN these are the major hyper parameters which we have tuned.

Epochs: Specified no of epochs helps the model to not being over-fit. For this we have created a function plot training history()which gives us 2 comparisons of Accuracy Curves and Loss Curves of Training and Validation Data. Visualizing them shows us that after certain no. of epoch validation accuracy is constant and validation loss starts increasing.So till that epochs we will be taking.
Batch Size As our samples are less(1560) in the dataset with no. of target ¿= 50. So for the small datasets the standard batch size to use is 32 or 64. So we have taken batch size either of them.
Neurons Density Initially we are taking very high density layers like (2048) but our data is not that much big so these dense layers is increasing the training time and also less accuracy than the moderate density layers(512 or 256 these too are highly dense) then converging it. So we are building the sequence with the moderate density neurons.
Number of Hidden Layers While having the more no. of hidden layers(6-7) has drastically decreased the mole's performance.So we just keep 2 hidden layers.This might be happened due to over-fitting of model and also increasing the computational complexity. Now we are training the ANN models separately on CNN,HoG,and lbp features. For that we have taken the same sequential layers the concatenated model have and then similarly we have hyper-tuned these above parameters as done above.
Also we have tried to club those 2 models one with the n of classes ¿= 50 and other with the n of classes between 10 to 50 to maximize the model preformance but we could not do that as there are 2 prediction made by 2 models on the test data, so we omitted that new thought.

D.3 Outcomes

We get the following outcomes after training the model other than the best model.

- Features when we have taken indivisually, HoG extracted features have best performance which is of **76.28 %**.
- From the other rest of the model with only CNN features and only lbp features the model with CNN has shown accuracy close to HoG of **71.15%** but the lbp have very less **34%** of accuracy.
- Graphs of Training accuracy v/s Validation accuracy and Training v/s Validation Loss can be seen.
- The table in this section shows the performance matrix of the best trained model(concatenated one) after hyper-parameter tuning.

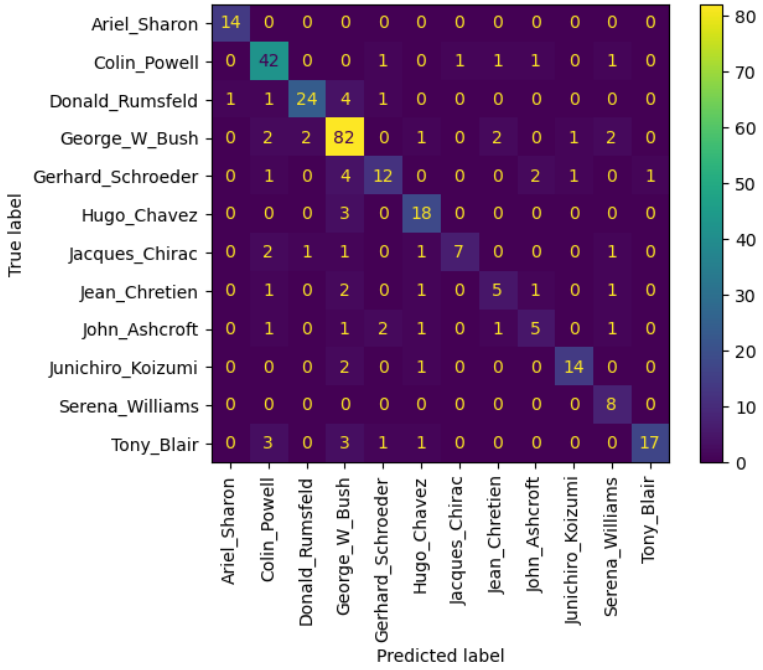


Figure 6: Confusion Matrix with Best ANN Model(Concatenated)

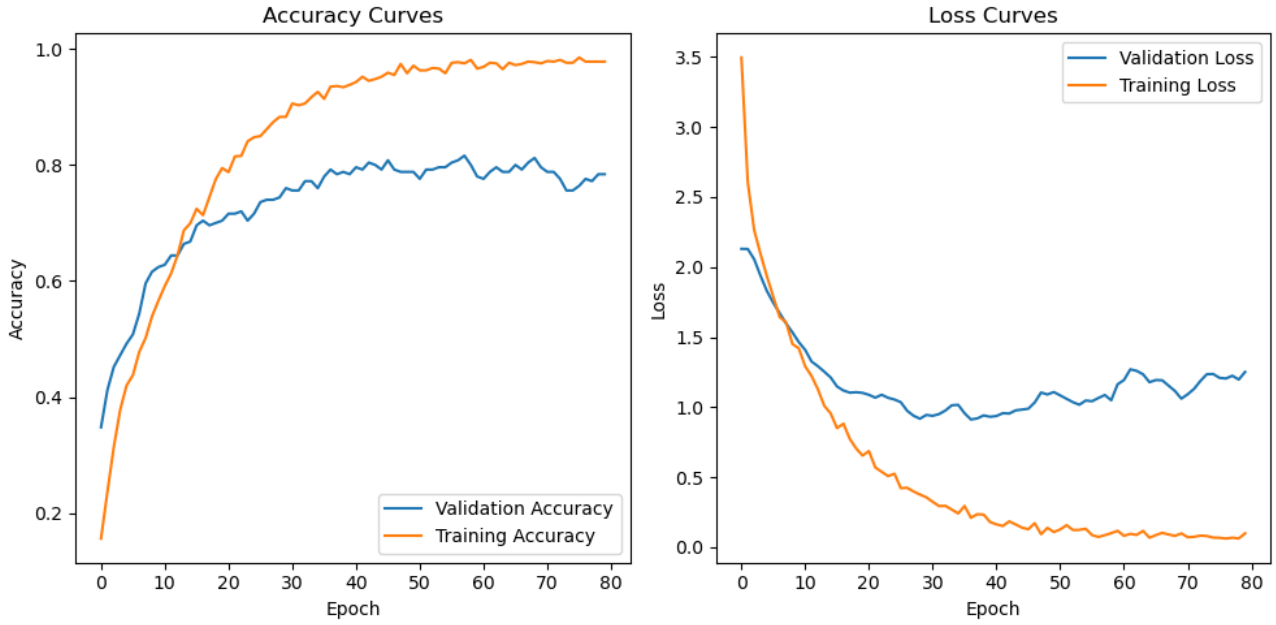


Figure 7: Accuracy Curves and Loss Curves Of The Best ANN Model(Concatenated)

Features	Precision	Recall	f1-Score	Accuracy
CNN	0.72	0.71	0.71	0.71
HoG	0.78	0.77	0.76	0.77
LBP	0.29	0.32	0.29	0.32
ALL	0.80	0.79	0.79	0.80

Table 4: Accuracy of Best Trained ANN Models

Features	Precision	Recall	f1-Score	Accuracy
CNN	0.72	0.70	0.70	0.70
HOG	0.76	0.76	0.75	0.76
LBP	0.25	0.25	0.24	0.25
ALL	0.88	0.88	0.87	0.88

Table 5: Classification Report of Best Model (LinearSVC)

E SVM (Support Vector Machine)

SVM is a supervised learning algorithm used for both classification and regression tasks. There are many kernels associated with SVM (i.e. Linear, Polynomial, RBF, etc.) that project the data into some other dimensional space so that it becomes better to plot a more accurate decision boundary hyperplane. Here, we are using it for the classification task.

E.1 Approach Used to train models

We have trained the LinearSVC; SVM with Linear, polynomial and RBF Kernel on the dataset of concatenated features(from CNN, HOG and LBP) and also on individual features that were extracted using CNN, HOG, and LBP.

E.2 Hyper Parameter tuning

SVM has the following important hyperparameters:

- C (for Linear, Polynomial and RBF Kernels)
- gamma (for Polynomial and RBF Kernels)

Both C and gamma have fixed ranges. C lies between 0 and 100, while gamma lies between 0 and 1.

C: Relates to the misclassifications that can be tolerated

- High C: Low Bias but potentially high variance (Overfitting)
- Low C: High Bias but low variance (Underfitting)

gamma: Determines the influence of individual training examples on the decision boundary.

- Low gamma: Decision Boundary goes in a general sense without bending close to individual points
- High gamma: Decision Boundary bends and twists to classify each training point correctly, which may capture noise and lead to overfitting.

We have trained the model and tuned these hyperparameters using grid search with param grid for C as [0.1, 1, 10, 100] and for gamma as [0.001, 0.01, 0.1, 1]. Then, after getting the best parameters from this, we performed a grid search again but with param grid containing the values of C and gamma in the vicinity of what we got earlier.

E.3 Outcomes

The best model for concatenated features is LinearSVC. Table 5 shows the performance metrics of this model on all the features.

The outcomes of other models are:

- Accuracies of SVM with Linear Kernel corresponding to the features:
  - ALL: 80.12%
  - CNN: 76.6%
  - HOG: 77.56%
  - LBP: 27.24%

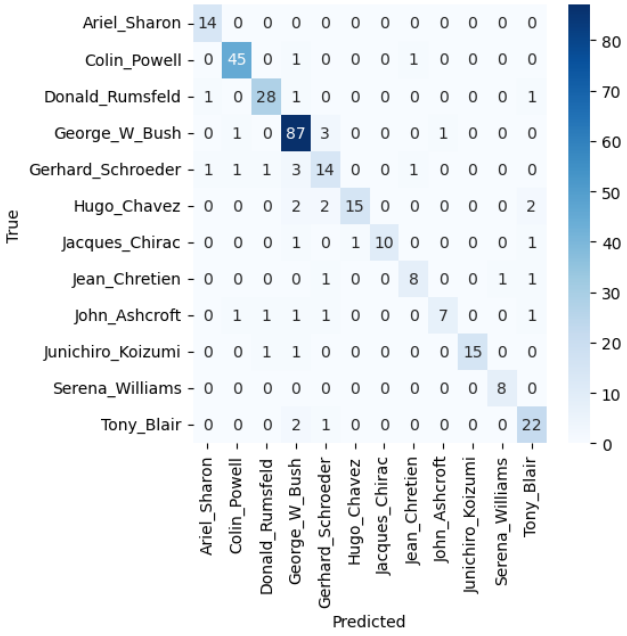


Figure 8: Confusion Matrix for LinearSVC model on concatenated features

- Accuracies of SVM with Polynomial Kernel with tuned hyperparameters corresponding to the features:
  - ALL: 30.45%
  - CNN: 37.18%
  - HOG: 29.49%
  - LBP: 32.69%
- Accuracies of SVM with RBF Kernel with tuned hyperparameters corresponding to the features:
  - ALL: 48.39%
  - CNN: 44.87%
  - HOG: 54.17%
  - LBP: 37.18%

E.4 Inference

- LinearSVC and SVM with Linear Kernel give the best accuracies both on concatenated features and individual features (CNN and HOG separately), which is almost close to 80%.
- LinearSVC on the concatenated features has the best accuracy of about 87.5%, which implies that the CNN extracts feature in such a way that they are linearly separable.
- The dataset of features extracted from LBP gave less accuracy(about 30%) in comparison to others because LBP features do not capture the complex structural information or variations in texture that are crucial for distinguishing between different faces.
- Accuracies of the polynomial and rbf kernel are less for all the datasets (concatenated features or individual) in comparison to LinearSVC and SVM with Linear Kernel as there might be the case of overfitting in the case of Polynomial and RBF Kernels as the accuracies on the training dataset after hyperparameter tuning is 100% for both.

F kNN Classifier

Experimental Setup: We conducted experiments to classify facial images using K nearest neighbour classifier. Three methods for extracting features were utilized: Histogram of Oriented Gradients (HOG), Convolutional Neural Networks (CNN), and Local Binary Patterns (LBP). We assessed the effectiveness of each feature extraction technique individually, as well as combined all three feature types. We also decided 2 parameters,

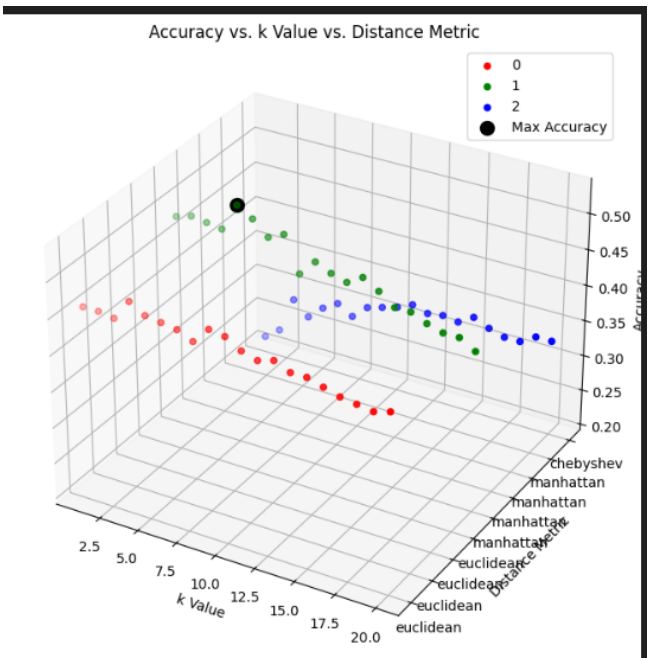


Figure 9: Graph showing optimized parameter of all the features combined

i.e.type of distance metric used(euclidean, manhatten and chebyshev) along with various values of "k"(number of nearest neighbours) varying from 1 to 20.

For two points, P = (n_1,n_2,n_3.....n_i) and Q = (m_1,m_2,m_3.....m_i) in i-dimensional space,

$$\textbf{Euclidean Distance} = \sqrt{(n_1 - m_1)^2 + (n_2 - m_1)^2 ... (n_i - m_i)^2}$$

$$\textbf{Manhatten Distance} = |n_1 - m_1| + |n_2 - m_2| ..... |n_i - m_i|$$

$$\textbf{Chebychev Distance} = \max(|n_1 - m_1|, |n_2 - m_2| .... |n_i - m_i|)$$

We performed KNN on the dataset where each class has number of instances to be greater than 50.We begin by splitting our dataset into training and testing sets in 80 : 20 ratio. Then we performed KNN by default setting of k=5 and taking distance metric as Euclidean distance. We proceeded to execute our model for every possible combination of both parameters to determine the optimized parameter combination that yields the highest accuracy.

Results:

Features used	Accuracy by default KNN parameters	Optimised parameters	Maximum Accuracy
HoG features	0.3945	euclidean,k=11	0.4871
CNN features	0.4647	euclidean,k=12	0.4711
LBP features	0.2564	manhattan,k=18	0.32051
all features	0.4743	manhattan,k=5	0.5256

Table 6: Comparison Table

The default parameters for KNN are squared euclidean distance as distance measuring metric and k=5. **Conclusion:** Combining all the features led to the highest accuracy, indicating that a diverse set of features can be beneficial for classification tasks. The choice of distance metric and the k value significantly influenced the performance of the KNN classifier. While optimization improved accuracy in some cases, it did not always result in substantial gains, suggesting that feature selection and engineering also play crucial roles in classification accuracy. Overall,we can say KNN performed moderately even after using optimized parameters for all of the features combined. So, it is not the best classifier for this job.

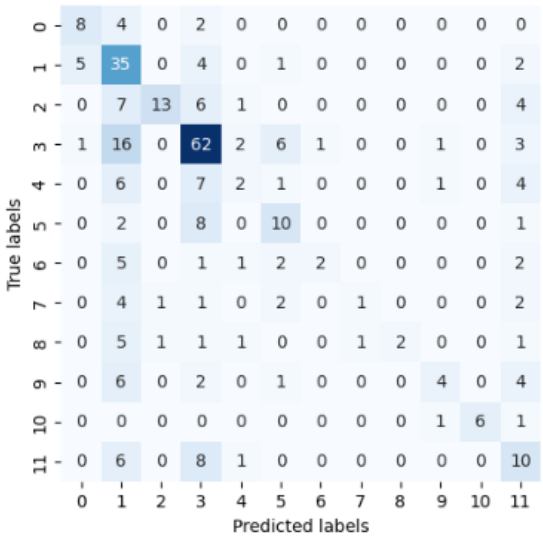


Figure 10: Confusion matrix for MultinomialNB on concatenated features

G Naive Bayes Classifier

G.1 Experimental Setup:

We conducted experiments to classify facial images using both GaussianNB and MultinomialNB classifiers. Three feature extraction techniques were employed: Histogram of Oriented Gradients (HOG), Convolutional Neural Networks (CNN), and Local Binary Patterns (LBP). We evaluated the performance of only HOG features, only CNN features, only LBP features, and then all features combined.

G.2 Using Gaussian Naive Bayes:

In Gaussian Naive Bayes, it's assumed that the features follow a Gaussian (normal) distribution. This makes it suitable for continuous features where the values can be modeled using a Gaussian distribution. GaussianNB calculates the likelihood of observing a certain feature value given a class label, considering the mean and variance of that feature for each class.

G.3 Using Multinomial Naive Bayes:

Multinomial Naive Bayes, on the other hand, is used when features follow a multinomial distribution. It's commonly employed for discrete features, especially in text classification tasks where features represent word counts or term frequencies. MultinomialNB calculates the likelihood of observing a certain feature given a class, considering the frequency of occurrence of that feature in the training data for that class.

G.4 Outcomes

- **Gaussian Naive Bayes:**
  - HOG: 64%
  - CNN: 33%
  - LBP: 16%
  - All features: 35%
- **Multinomial Naive Bayes:**
  - HOG: 30%
  - CNN: 49%
  - LBP: 30%
  - All features: 50%

G.5 Inferences:

- Multinomial Naive Bayes outperformed Gaussian Naive Bayes in our facial recognition task.

Features	Precision	Recall	f1-Score	Accuracy
HOG	0.11	0.09	0.05	0.30
CNN	0.58	0.40	0.41	0.49
LBP	0.09	0.10	0.07	0.30
ALL	0.59	0.40	0.42	0.50

Table 7: Classification Report of Best Model (MultinomialNB)

- This superiority is attributed to Multinomial Naive Bayes' effectiveness in handling discrete features, leading to higher accuracy across various feature extraction techniques.
- Gaussian Naive Bayes, designed for continuous features, faced challenges in capturing the underlying distribution of discrete features, resulting in lower accuracy.
- Therefore, for tasks involving discrete feature representations like those derived from HOG, LBP, and combined features, Multinomial Naive Bayes is the preferred choice due to its superior performance.

VI. Ensemble of Classifiers

- We created an ensemble of all the models trained
- This was done to get the maximum of all the models
- We are importing the saved models through joblib, a python library
- Then, we are predicting the class given the image
- We take weighted predictions of each model based on their accuracies
- The model's prediction is added the weight number of times into a list
- Finally, we declare the prediction as the most frequent element of the list
- Further in this we made a similarity predictor between two images using ANN:

A Similarity Calculation Method

Feature Extraction: Initially, the features of both images are extracted using a concatenation process, followed by reshaping them for compatibility with the model.

Prediction: The extracted features are then fed into a pre-trained ANN for prediction. The model predicts the class probabilities for each image.

Comparison: The argmax values of the predictions for both images are compared. If they are the same, it implies that both images likely belong to the same class. In this case, the mean probabilities of the classes are computed as a measure of similarity.

Divergence Calculation: If the argmax values differ, indicating potential differences in the classes, the Kullback-Leibler (KL) divergence between the predicted probability distributions of the two images is calculated. This divergence serves as a measure of dissimilarity between the images.

Output: The method returns the similarity score along with a label indicating whether the images are considered "Same" or "Different" based on the calculated metric.

VII. Inferences

Using all models, the following inferences were drawn:

- Face Recognition is a complex data set, wherein traditional Machine Learning techniques may not work if the data set is not clean.
- The models couldn't give good accuracy because of probably one single class George W Bush.

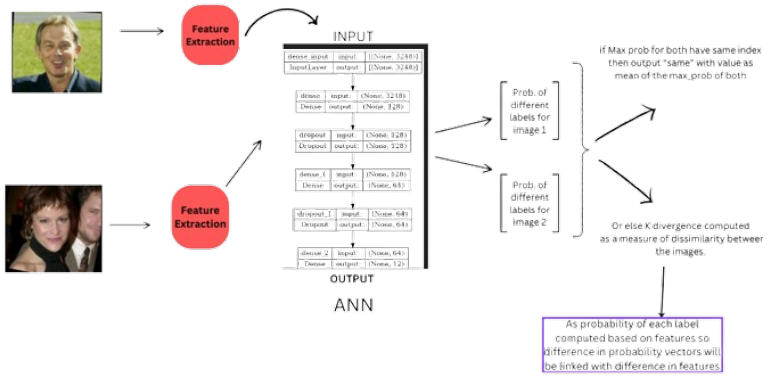


Figure 11: Our Proposed Similarity Predictor

- We could infer that the above said class has relatively very high number of images, thus confusion most of the models and getting them biased towards it.
- resnet50 extracted features are the best in majority of the models, since they are able to capture the textures of the face, which helps in better distinction between images.
- We tried creating an ensemble of Random Forest hoping for very good results, but the outcome was opposite. The data is biased towards single or two classes. So, all the Random Forests too got trained onto the same data set. So, all the random forests predicted the wrong class. This wrong prediction got amplified as the confidence level of the wrong prediction increased due to other Random Forests predicting the same wrong class.
- Artificial Neural Networks did work well on this data, giving an accuracy of about 80 % . However, we noticed that the values of hyperparameter follow a Gaussian type of distribution.

VIII. Summary

In this project, our primary objective was to improve the accuracy of facial recognition systems. To achieve this goal, we explored a variety of feature extraction techniques and classification models. Specifically, we focused on Histogram of Oriented Gradients (HOG), Convolutional Neural Networks (CNN), and Local Binary Patterns (LBP) as feature extraction methods. These techniques are renowned for their ability to capture distinctive facial features essential for precise identification.

Following feature extraction, we embark on an extensive evaluation of different classifiers to predict facial identities. Our toolkit includes Support Vector Machines (SVM), k-Nearest Neighbors (k-NN), Random Forests (RF), Naive Bayes classifier, Artificial Neural Network (ANN), Ensemble Learning, and decision tree. Through rigorous experimentation, we meticulously optimize hyperparameters to maximize classification accuracy. This report summarizes our experience, discoveries, and suggestions for improving facial recognition technology.

IX. Contributions Of Each Member

- **Swaksh Patwari(B22AI065)** Artificial Neural Network (ANN)
- **Aditya Rathor(B22AI044)** Feature Extraction , Decision Tree
- **Sumeet S Patil(B22CS052)** Random forest and Random Landscape
- **Manya (B22CS032)** Naive Bayes Classifier
- **Avani Rai (B22CS094)** K-nearest neighbour Classifier(kNN)
- **Om Kumar (B22CS081)** Support Vector machine(SVM) with various kernels