Photo Cluster: Automated Face Recognition and Clustering for Image Grouping

Veda Kailasam, Jorge Chavez , Philmon Roberts , Paul Osuma
University of Illinois Urbana- Champaign
Champaign, Illinois, USA
vedak2@illinois.edu, jorgejc2@illinois.edu, philmon2@illinois.edu, posuma2@illinois.edu

1 INTRODUCTION

With the exponential growth of digital image repositories, organizing and managing vast collections of photos have become increasingly challenging. This paper introduces "Photo Cluster," a cutting-edge system designed to address this issue by implementing automated face recognition and clustering techniques for efficient image grouping. Leveraging state-of-the-art facial recognition algorithms, Photo Cluster accurately identifies individuals within photos and organizes them into meaningful clusters based on shared faces. The integration of advanced machine learning and computer vision technologies enhances the speed and precision of face recognition, optimizing the overall organization and accessibility of image collections. Through rigorous experimentation on diverse datasets, the efficacy of Photo Cluster is demonstrated, showcasing its potential to revolutionize how users interact with and manage their visual archives.

In our approach we begin by detecting a face given an input image. We then generate embeddings for the face before finally using this characteristic information to find similar faces in a pre-built dataset. The output is a cluster of faces belonging to the same facial identity of a person.

2 METHODOLOGY

The flow chart in the second column shows the hierarchy of our photo clustering pipeline. Our system takes as input portrait images as well as group photos, channels them through a feature extraction stage before clustering is finally performed using a pre-selected dataset.

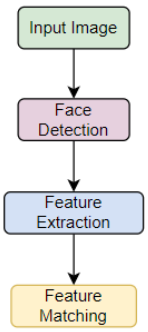


Fig. 1: General Vision Pipeline

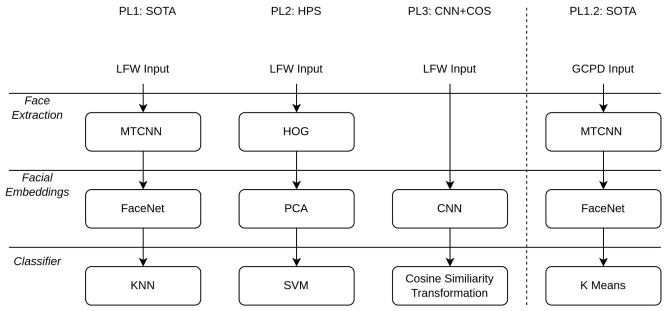


Fig 2: 3 Deployed Pipelines

More distinctly, we use the 3 pipelines shown in fig. 2 and describe their accuracies as well as the methodology behind each component. Throughout the rest of the paper, we will mention which pipeline we are referring to by their abbreviations: PL1, PL2, PL3, and PL1.2. Notice that PL1, PL2, and PL3 are used on the LFW input dataset. For these three pipelines, supervised learning is conducted to directly compare the accuracy results, but PL1.2's pipeline is simply empirically observed on the family group images found in the GCPD dataset. PL1.2 is included to show that PL1 can still demonstrate very good results on unmarked and group image based applications, showcasing our true Photo Cluster implementation.

2.1 Dataset

The dataset curated for the Photo Cluster project comprises a diverse collection of facial images sourced from various environments and demographics. Encompassing a broad spectrum of facial expressions, lighting conditions, and backgrounds, the dataset aims to challenge the models with real-world complexities.

2.1.1 LFW

Labeled Faces in the Wild (LFW) is a publicly available dataset used as a benchmark for face recognition. It comprises 13,000 images of faces collected from the web. 1680 of the persons have at least two photos present in the dataset. The name of each person is included in the dataset as the label for the image. Virtually all images consist of a single face. The few images that have multiple faces are labeled with the name of the person with the largest face found in the image. Since this dataset is labeled, this makes it ideal for performing supervised learning and collating relevant metrics. In addition, due to the pervasive use of

LFW in computer vision research, we are able to evaluate different stages of the pipeline with state of the art (SOTA) results. There are four versions of the LFW dataset but we employ the *deep funneled images* variant, which has been shown to produce the best results for face detection and recognition algorithms.

2.1.2 GCPD

The Gallagher Collection Person Dataset (GCPD) is a dataset composed of images similar to what one would expect from a photo gallery. It consists of 589 images of 32 unique persons, capturing the physical environment, facial expressions, and common events likely to be included in a photo album. The dataset is partially labeled — only one of the faces in each image is labeled. Since most images include multiple people, we treat this as an unlabeled dataset and perform evaluations using the limited labels. GCPD provides different benefits from the LFW dataset, which is used to benchmark results at several points in the pipeline. GCPD, on the other hand, enables us to extend our use case to permit users to upload their photo gallery and a chosen image, and receive all the images in the gallery where the persons in the chosen image are found. While a model learned on this dataset will likely need to be retrained for another dataset provided by a user, we expect similar results. In general, we assume that the GCPD dataset is drawn from a distribution of personal photo albums, where the user's input is also a sample from the same distribution.

2.2 Face Detection

The face detection component of this project focuses on developing and refining algorithms to accurately identify and localize faces within images. Leveraging a meticulously curated dataset that spans diverse scenarios, lighting conditions, and demographics, the project aims to create a robust face detection model. Various techniques, including HOG, MTCNN, and potentially others, will be explored for their effectiveness in detecting faces amidst challenging conditions such as occlusion and varied expressions.

2.2.1 Histogram of Gradients

In an effort to apply the fundamental computer vision techniques we learnt in class, we implemented our own face detector from scratch using Histogram of Oriented Gradients (HOG). Faces tend to have a characteristic pattern for the orientation of edges found in them and this is what is exploited by the face detection model to pinpoint where these 'face-like' features are located in an input image.

To generate a HOG descriptor over a patch of an input image, the following procedure is used. Horizontal and vertical gradients are generated from the patch by filtering it with opencv's sobel gradient filter. The gradients are then converted from cartesian to polar form so that we end up with their magnitude and orientation. A 9 bin histogram is then used to package the orientations, weighted by their corresponding magnitudes and this becomes the feature vector for that patch. A patch is divided into 8 by 8 grid cells so we end up with 64 such HOG feature vectors. The reason for dividing a patch of an image into 8 by 8 grid cells is so that the patch is less sensitive to noise.

Mitigation strategies for corruption of gradient results also extend to accounting for lighting changes.To make the HOG descriptors independent of lighting variations, the 8 by 8 grid cells are normalized using 16 by 16 block normalization. In this process, four 8 by 8 grid cells are merged into one block and each cell's value is divided by the norm for all cells within the 16 by 16 block. The output is our final HOG descriptor for a given patch of an image.

In order to distinguish 'face-like' patches from non-face like patches we trained a model using a Linear Support Vector machine to do this. We obtained our positive data of faces from lfw-dataset and negative data of non-faces from skimage library. After improving our model using sklearn's GridSearchCV it ended up having a best score of 0.9906. When patches of an test image generated using a sliding window function are fed into this model, the model not only returns the labels for every patch as being a face or not but also the the decision score for the every patch. We leveraged this decision score for our non maximum suppression step by filtering out the highest scoring positive face label as the face region and drew a bounding box around it.

To achieve face detection at multiple scales, we wrote a function to run the sliding window on an input image at 3 scales; 1, 2 and 3. The best scale was then used to draw the bounding box. Below are examples of the outputs of the HOG face detector model. Our model performs well for face images of different scales, for both frontal and slightly non-frontal faces and for slightly occluded faces, demonstrating the robustness of the HOG as a feature descriptor.

Fig 3: Face detection at scale 1.0. Face dimension 62 x 47 pixels

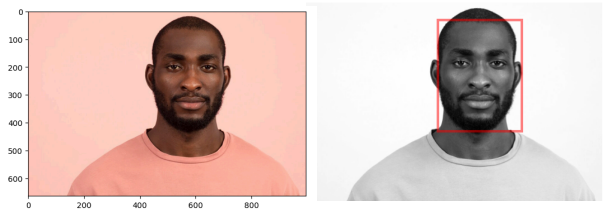


Fig 4: Face detection at scale 3.0. Face dimension 186 x 141 pixels

Fig 5: Face detection for non-frontal face. Scale = 3.0

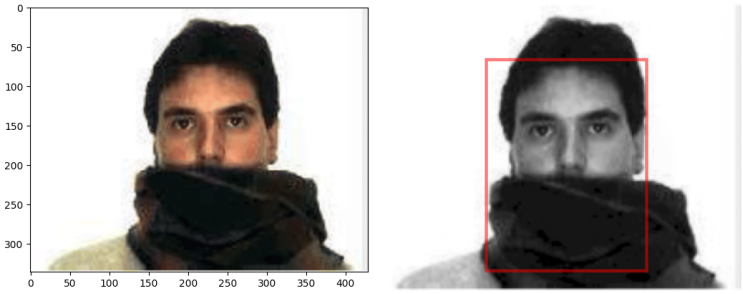


Fig 6: Face detection under occlusion. Scale = 2.0

2.2.2 MTCNN

The state of the art (SOTA) face detector MTCNN v2.5.3 is used to detect faces in both the LFW dataset and the GCPD dataset. MTCNN takes an image as input and returns possibly one or more detected faces as a 160 * 160 pixel image. In particular, MTCNN allows for robust detection in the face of occlusion and unaligned images. We use MTCNN as a ground truth for evaluating the results of our HOG face detector on the LFW dataset. We limit our comparison with the LFW dataset since the HOG detector only extracts one face, even if there are several faces present in the image. MTCNN, on the other hand, permits the detection of multiple faces in an image. Consequently, we deploy MTCNN on the GCPD dataset extracting multiple faces per gallery image for further processing later in the pipeline.

Below is an example of MTCNN used on the GCPD dataset for face detection:

Fig 7: MTCNN Face detection on GCPD dataset

2.3 Feature Extraction

In the context of this project, the feature extraction phase plays a pivotal role in distilling essential facial characteristics for subsequent face recognition and clustering tasks. The project employs a multi-faceted approach encompassing techniques such as SVM with PCA for dimensionality reduction and CNNs for hierarchical feature learning. By leveraging these methods, the project aims to extract discriminative facial features that transcend variations in expressions, poses, and lighting conditions. The objective is to enhance the system's ability to accurately represent and distinguish unique facial attributes, ultimately contributing to the robustness and efficacy of the overall face recognition and clustering pipeline.

2.3.1 FaceNet

FaceNet is a leading facial recognition system developed by Google researchers. It overcomes the challenges associated with facial recognition by employing a high-dimensional embedding space for facial features. It offers robust facial representation that performs in spite of variations in lighting, pose, and expression. We therefore use FaceNet as a benchmark to generate premier embeddings for both LFW and GCPD dataset.

2.3.2 PCA

For feature extraction, the images detected above were trimmed, resized, and used to construct an embedding. These embeddings were curated using the Principle Component Analysis (PCA) method. It is ideal to use an embedding of the preprocessed image as it can reduce dimensionality while preserving the most significant

features before training/predicting its label using some classifier.

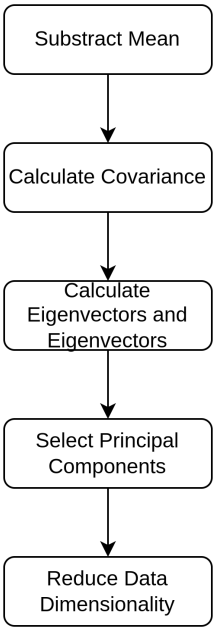


Fig 8: PCA Dimensionality Reduction

There are two main benefits: reducing noise on the image and computationally speeding up the training process. This was especially true with using SVC as the classifier which could take some time running on embeddings for the entire dataset (~5 minutes on CPU, ~25 seconds on GPU).

The PCA methodology involves first subtracting the mean from all the data points so that there is zero mean on the dataset. Next is calculating the covariance matrix on all the random variables. This provides information as to the linear dependencies on any two random variables. The next step involves calculating the eigenvectors and eigenvalues of the data. They are calculated using the covariance matrix, and represent data with higher variance.

Their respective eigenvalues determine the data set's variance. The fourth step is selecting the principal components. From the available eigenvectors, we only choose a select few to keep. The criterion for choosing which eigenvectors to hold onto is determined by the relative variance associated with the eigenvectors. If an eigenvector can explain most of the data's variance, we choose to keep it. If it explains a more insignificant amount, it is discarded. For facial embeddings, these principal components can be visualized and they are referred to as eigenfaces (Fig. 9). Naturally, a lot of fine grain detail is removed from the original faces, but the structure is kept. The final step which involves data dimensionality reduction simply involves projecting the data onto the selecting principal components. For demonstration, this was done on a subset of 7 faces using only two principal components in figure 10. For deployment, 128 components were used to curate our results with the SVM classifier. With this, we have an embedding with less dimensions and can hopefully characterize most of the variance found in a set of images.

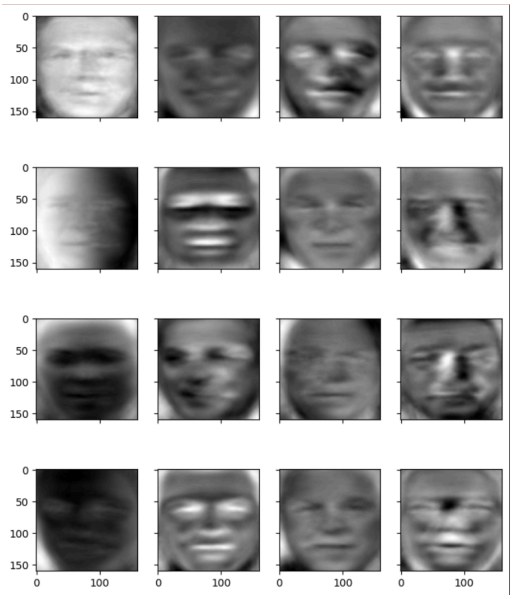


Fig. 9: Eigenfaces on a subset of LFW

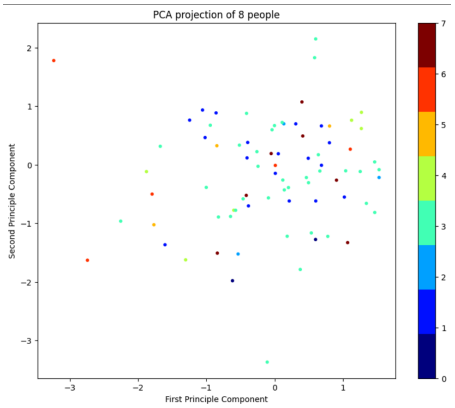


Fig. 10: Two Component Projection on LFW Subset

2.3.3 Face Recognition with CNN

We propose a methodology for generating face embeddings using Convolutional Neural Networks (CNN) on the SKlearn Labeled Faces in the Wild (LFW) dataset. We begin by preparing the dataset, conducting face detection, and normalizing facial images. The CNN architecture is designed with convolutional layers, activation functions, pooling layers, and fully connected layers to capture intricate facial features. The model is trained on the labeled training set, and face embeddings are extracted from the penultimate layer during testing. Evaluation metrics, including accuracy and precision, are employed to assess recognition performance. Fine-tuning is considered based on evaluation results, and a

comparison with baseline methods is conducted.

Fig 11: Faces from the LFW Dataset

The model is a neural network architecture that is defined using the Keras Sequential Model (Fig 7). The first layer is a 2D convolutional layer with 32 filters, a kernel size of (3, 3), and a linear activation function. Leaky Rectified Linear Unit (LeakyReLU) activation is applied, followed by max-pooling with a (2, 2) pool size and a dropout layer with a rate of 0.25. This pattern is repeated with two more convolutional blocks, each doubling the number of filters (64 and 128) and adjusting dropout rates. The final layers consist of flattening the output, two densely connected layers with 128 and 64 neurons, LeakyReLU activation, and dropout, and an output layer with softmax activation for the specified number of classes.

The training data is fed to the model for a specified number of epochs (60 in this case), with a batch size of 100. The training process is monitored for improvements in loss, and the training is validated using a 10% split from the training data. An early stopping callback is implemented, which halts training if no improvement in loss is observed for 10 consecutive epochs.

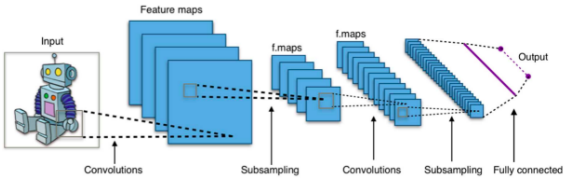


Fig. 12: CNN Breakdown

The convolutional neurons respond to a portion of the input from the previous layer (called the local receptive field, with overlap between the regions), extracting higher-level features of the input; the neurons of the pooled layer are input to the previous layer

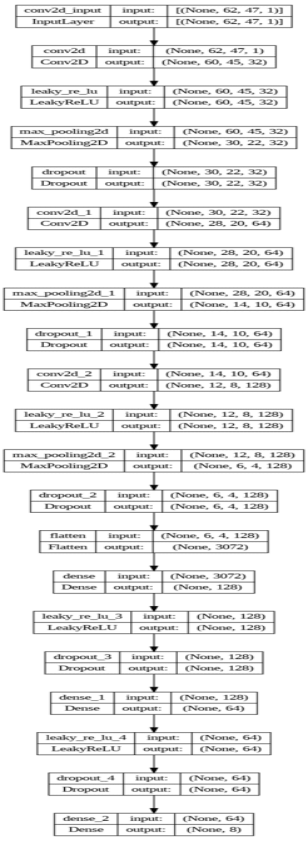


Fig 13: CNN Model Architecture

The classification report reveals that the trained convolutional neural network demonstrates strong performance in facial recognition, achieving an impressive overall accuracy of 94% on the test dataset. Precision values, representing the accuracy of positive predictions, are consistently high across different classes.

	precision	recall	f1-score	support
Ariel Sharon	0.79	0.73	0.76	15
Colin Powell	0.99	0.97	0.98	68
Donald Rumsfeld	0.90	0.84	0.87	31
George W Bush	0.95	0.98	0.96	126
Gerhard Schroeder	0.84	0.91	0.87	23
Hugo Chavez	0.90	0.90	0.90	20
Junichiro Koizumi	1.00	1.00	1.00	12
Tony Blair	0.97	0.93	0.95	42
accuracy			0.94	337
macro avg	0.92	0.91	0.91	337
weighted avg	0.94	0.94	0.94	337

Fig 14: Classification Report

2.3.4 Face Recognition with InceptionResNetV2 (PL1)

Developed a face recognition system by combining the power of the MTCNN (Multi-Task Cascaded Convolutional Networks) for face detection and FaceNet for face embedding. This methodology involves leveraging pre-trained models to perform face detection and extraction of facial embeddings. MTCNN model is employed for accurate face detection within images. Images from the dataset are processed through MTCNN to identify and extract faces, ensuring precise localization through bounding box coordinates. The FaceNet model,

specifically InceptionResNetV2, is utilized for face embedding. The model generates high-dimensional feature vectors representing facial characteristics. A custom function is implemented to extract and flatten these embeddings for further analysis. Cosine similarity is computed between the face embeddings of the testing and training sets. This metric serves as a measure of similarity between the facial features, forming the basis for subsequent predictions. The model predicts face labels based on the highest cosine similarity, effectively recognizing faces within the testing set. The argmax function is employed to determine the predicted labels. The system generates visual representations of the face recognition results by randomly selecting faces from the testing set.

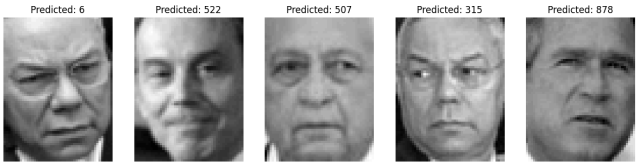


Fig 15: Predictions from InceptionResNetV2

2.4 Feature Matching

After obtaining the face embeddings from the specific models, the faces are clustered or grouped based on similarity metrics. These approaches include K-Means clustering, k-Nearest Neighbors (kNN), Support Vector Machines (SVM). These embeddings serve as feature vectors that encapsulate distinctive facial characteristics. If using cosine similarity, the embeddings are subjected to a similarity analysis to measure the likeness between pairs of faces, providing a continuous metric for resemblance. Alternatively, K-Means clustering is applied to group similar faces together based on their embeddings, revealing natural clusters within the dataset. In the case of kNN, the embeddings are utilized to identify the nearest neighbors for each face, facilitating efficient face retrieval. Evaluation metrics, such as precision, recall, and accuracy, are then employed to assess the performance of the chosen method (cosine similarity, K-Means, or kNN) in capturing meaningful facial similarities. Visualization techniques aid in interpreting the distribution of embeddings or clusters, providing insights into the relationships between faces.

2.4.1 Feature Matching with Cosine Similarity

In the process of feature matching using cosine similarity for face embeddings, we consider the embeddings that we obtained from one of the trained face recognition models. These embeddings serve as high-dimensional vectors that encapsulate the distinctive facial features essential for subsequent matching. Following this, pairwise cosine similarity calculations are performed between the obtained face embeddings. The cosine similarity metric, which ranges from -1 (indicating complete dissimilarity) to 1 (representing complete similarity), quantifies the resemblance between pairs of faces based on both direction and magnitude of their vectors. The subsequent decision-making step involves either setting a similarity threshold or ranking the faces based on their cosine similarity scores. The threshold establishes the minimum similarity required to consider two faces a match, while ranking provides an ordered list of faces based on their level of resemblance. With these criteria in place, a matching decision is executed for each pair of faces. If the cosine similarity exceeds the set threshold, the faces are deemed a match. Alternatively, the ranking is utilized to identify the most similar faces in the dataset. We calculate the cosine similarity between the test sample embedding and all training embeddings i.e. the embeddings of all the faces in the database, and then determine the indices of the top k most similar training samples. The subsequent visualization step provides a qualitative understanding of the similarity relationships between the test sample and its top matches in the training set.

2.4.2 Feature Matching with K-Means

With facial embeddings in hand, K-Means clustering is applied to group similar faces together based on their learned features. The K-Means algorithm aims to partition the dataset into K clusters, where each cluster represents a group of faces with similar characteristics. The number of clusters (K) can be specified beforehand or determined using optimization techniques such as the elbow method. After clustering, each face is assigned to the cluster with the nearest centroid, allowing for the identification of natural groupings within the dataset.

In the context of facial embeddings, t-SNE can be applied to reduce the dimensionality of the embeddings while preserving their pairwise similarities. This enables the creation of a 2D or 3D plot where each point represents a facial embedding, and the proximity of points reflects the similarity between corresponding faces. Clusters of similar faces tend to be visually grouped together, providing an intuitive representation of the learned feature space. Faces with similar features are positioned close to each other, revealing patterns and relationships that might not be apparent in the original high-dimensional space. This visualization aids in interpreting the effectiveness of the K-Means clustering algorithm in capturing meaningful facial similarities.

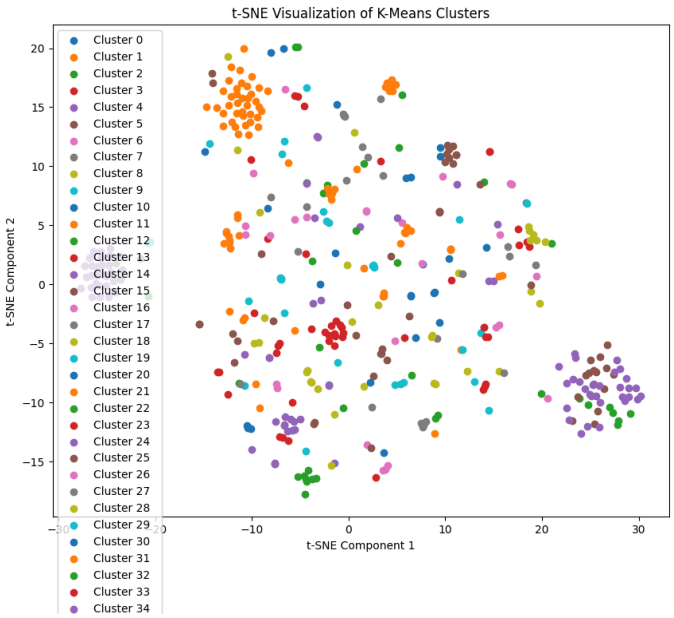


Fig 16: TSNE Plot on LFW

2.4.2 Feature matching with Support Vector Machines

Support vector machines or SVMs are supervised learning techniques used to classify and perform regression analysis on data. There are various kernels associated that can be used in the algorithm including the standard linear kernel, polynomial, radial basis function (RBF), and sigmoid kernels. For facial recognition, the more popular kernel to use is the RBF kernel, but the linear kernel was used in our results. It is also typically paired with PCA for data embeddings whose results we demonstrate as well, but it is compatible with other data reduction embeddings such as Facenet. The main attraction of SVM is how extremely efficient it is to run, making it ideal for real time and/or mobile applications. From our results, we find that it is robust under a more uniform subset of the LFW dataset, showcasing the fascination behind the algorithm, but can quickly fall apart when being used on the full, sparse dataset.

For training, a linear regression is done on the labeled data, and linear regression is performed so that features extracted from the embeddings can be mapped to a target. Across each dimension in the embedding, a decision boundary is created and altered in the training process. The goal of SVM is to draw these decision boundaries such that they maximize the distance to support vectors. Once trained, new inputs are mapped into this high dimensional space and their placement with respect to these boundaries determine the target label it is associated with.

3 RESULTS

3.1 CNN based face matching (PL3)

We are using Sklearn's LFW dataset where we are considering 60 images per person from the dataset. The dataset is tailored to include individuals with a sufficiently varied collection of facial images. There are 1348 samples, each comprising images of dimensions 62x47 pixels. Subsequently, the dataset is divided into training and testing sets for evaluating the model's performance on unseen data. Here, 25% of the dataset is allocated to the testing set (test_size=0.25), and a fixed random seed (random_state=42) is set for reproducibility.

The model's performance on the test dataset was evaluated, revealing a notable accuracy of 93.7%. This implies that the model correctly classified the majority of instances in the test set. Additionally, the calculated loss on the test dataset was found to be 0.2, indicating a relatively low level of misclassification. These results collectively underscore the effectiveness of the convolutional neural network in accurately predicting the target classes for the given images. The high accuracy and low loss values suggest that the model has successfully learned and generalized patterns from the training data, demonstrating its robustness in face recognition tasks.

```
11/11 [==============================] - 1s 42ms/step
The accuracy of the predicted value from the model: 97.94261381030083
11/11 [==============================] - 1s 38ms/step - loss: 0.2672 - accuracy: 0.9377
The loss value of the test dataset 0.2671946883201599
The accuracy value of the test dataset: 93.7685489654541
```

Fig 17: Accuracy of CNN model

The classification report reveals that the trained convolutional neural network demonstrates strong performance in facial recognition, achieving an impressive overall accuracy of 94% on the test dataset. Precision values, representing the accuracy of positive predictions, are consistently high across different classes, the recall values, indicating the model's ability to capture all instances of a class, are also commendable and The F1-score, balancing precision and recall, reaffirms the model's robustness. Overall, the classification report attests to the effectiveness of the convolutional neural network in accurately recognizing faces, demonstrating its proficiency across diverse classes in the given dataset.

The below graphical representation presents transparency and facilitates qualitative evaluations of the model's facial recognition performance on a per-instance basis.

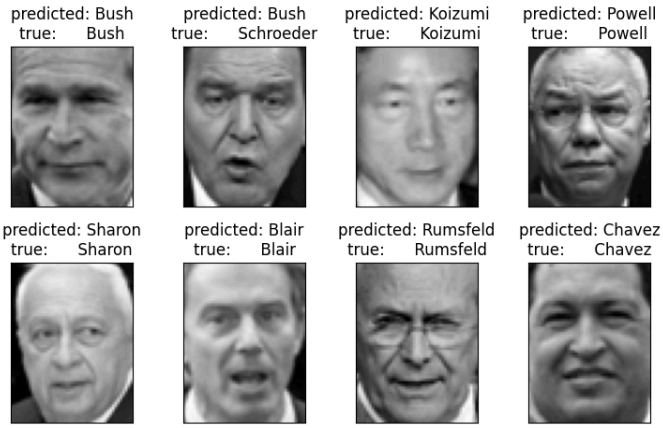


Fig 18: CNN Model Predictions

For monitoring the model's learning process and making necessary adjustments to optimize training, we visualize training and validation metrics during the training of the neural network model. Fig. 19 and Fig. 20 illustrate two plots where the first plot is dedicated to displaying the training and validation loss and the second plot illustrates the training and validation accuracy.

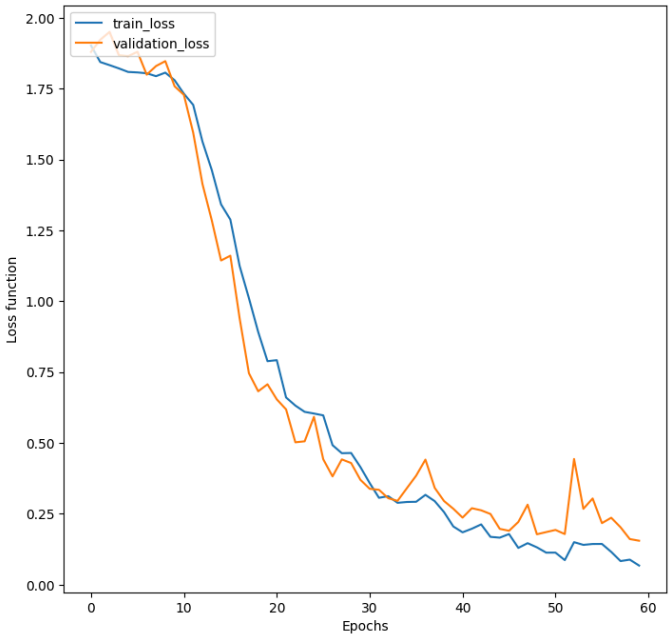


Fig 19: Epoch-Loss tradeoff

The training loss is depicted by the line graph labeled 'train_loss,' and the validation loss is represented by the line labeled 'validation_loss.' The x-axis denotes the number of epochs, reflecting the training iterations, while the y-axis corresponds to the values of the loss function.

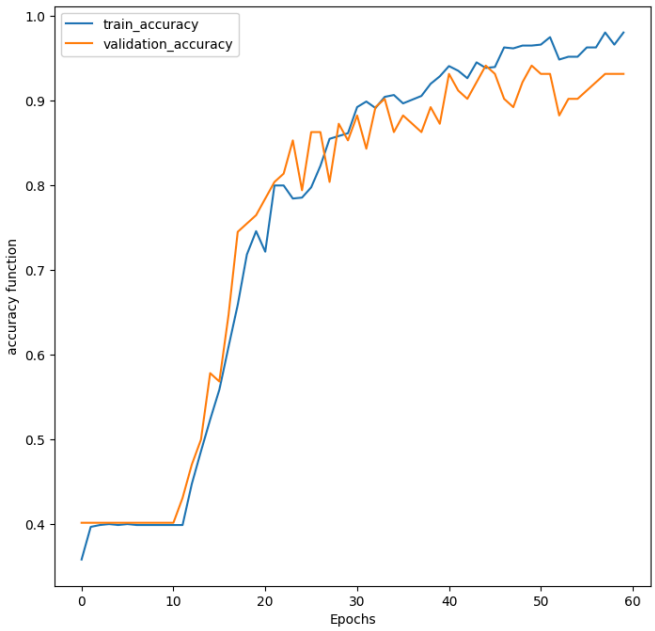


Fig 20: Epoch-Accuracy tradeoff

The training accuracy is plotted as 'train_accuracy,' and the validation accuracy is plotted as 'validation_accuracy.' Similar to the first subplot, the x-axis signifies the epochs, and the y-axis represents the accuracy values. A legend is included to differentiate between the training and validation accuracy curves.

3.2 SVM Face Based Matching (PL2)

To first compare our SVM approach to the CNN based matching, we once again use the subset of LFW on persons with at least 60 images. We then extend our discussion to the structure of the dataset and its effect on the SVM accuracy.

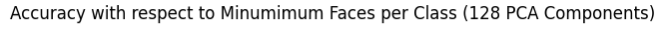
We get a fair accuracy of 81% (Fig. 23) which demonstrates that SVM's accuracy does not trail far behind the CNN approach. Fig. 21 demonstrates a test on the same input images used but after being run through our HOG facial extractor. (discuss the empirical results once the table is ready)

Next, we demonstrate how the accuracy is affected by including more of the dataset in Fig. 23. This table displays the accuracy of the SVM on its training set and testing set. The rows demonstrate the hyperparameter we chose to vary, which is the minimum number of images a class needs to have in the dataset to be considered in that training/testing run. Naturally, a minimum number of 1 means that the entire dataset was included. The large and sparse nature of the LFW dataset makes it clear that SVM is not suited for datasets of this kind, any may be far more suited for more uniform datasets.

Fig. 21: SVM Model Predictions

	Count
Ariel_Sharon	59
Colin_Powell	172
Donald_Rumsfeld	88
George_W_Bush	406
Gerhard_Schroeder	86
Hugo_Chavez	51
Junichiro_Koizumi	44
Tony_Blair	105

Fig 22. Sample Count from LFW Subset



	Train Accuracy	Test Accuracy
Min Faces 60, Classes 8	97.13 %	81.60 %
Min Faces 50, Classes 12	96.41 %	81.54 %
Min Faces 40, Classes 19	95.64 %	73.66 %
Min Faces 30, Classes 34	95.10 %	65.60 %
Min Faces 20, Classes 62	93.69 %	59.26 %
Min Faces 10, Classes 158	91.52 %	49.21 %
Min Faces 5, Classes 423	88.95 %	36.81 %
Min Faces 1, Classes 5749	52.71 %	14.66 %

Fig 23. Accuracy using SVM with varying minimum images per face

3.3 KNN based feature matching (PL1)

K-nearest neighbors (KNN) was used on the subset of LFW dataset (8 classes) to classify faces based on the class of the closest feature using the euclidean distance metric. The hyperparameter K was set to 1, as it achieved the lowest error rate on the validation set for suitable values of K. We obtain testing accuracy of 97.8% with this configuration. Since KNN achieved the best results of all the methods on the subset of LFW, we were curious to evaluate this pipeline on the entire LFW dataset. A testing accuracy of 88.3% was achieved, which signaled the model generalized well.

3.4 K-means based feature matching (PL1.2)

We evaluate the K-means model (*K=32, random_state=5*) on the GCPD unlabelled dataset. To visually observe how well K-means perform, we leverage TSNE for dimension reduction before plotting the corresponding graphs.

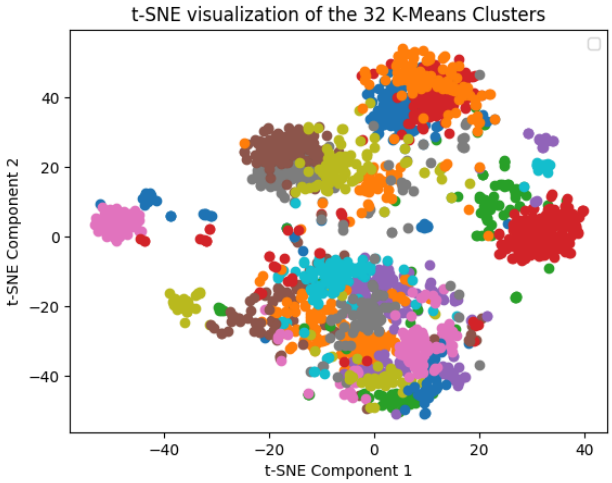


Fig 24: TSNE Plot on GCPD

Since the ground truths (the labels of all faces found in the image) are not available, we utilize the silhouette score and inertia to provide a reasonable measure of K-means performance. A silhouette score of 0.07 and an inertia of 777.45 was achieved.

4 DISCUSSION AND CONCLUSIONS

In this project, several key insights were gained through analysis and experiments conducted. The main approaches included face detection using HOG and MTCNN, SVM trained with PCA for face embedding extraction, building a CNN from scratch, utilizing a pre-trained InceptionResNetV2 model, and matching through cosine similarity, k-means, and k-nearest neighbors (KNN).

Main Insights:

1. Face Detection Techniques: HOG and MTCNN were employed for face detection, showcasing the importance of robust techniques in accurately identifying faces within images. Our implemented HOG face detection proved to work well for different scales of faces, occluded faces and slightly non-front facing faces.
2. Feature Extraction and Detection with SVM and PCA: SVM trained with PCA was utilized for face embedding extraction. This approach aimed to capture the essential facial features for subsequent recognition and clustering. This technique is known for being attractive towards real time applications, but as shown from our results, performed poorly on the entirety of the LFW dataset. This makes clearer sense taking into consideration the large sparsity and class imbalance in the full dataset. Sparsity in the sense that most faces only had one image to their name. Class imbalance in the sense that some faces only had a few images, and some faces such as George W. Bush had

523 images associated with them. The sparsity diminishes the amount of structure that PCA can leverage since its effectiveness relies on capturing variance in the data. The class imbalance also proves challenging for SVMs as the algorithm is known to be quite sensitive to balanced datasets as it can be particularly biased towards the majority classes.

3. Convolutional Neural Network (CNN): A CNN was implemented both from scratch and using a pre-trained InceptionResNetV2 model. CNNs demonstrated superior performance compared to the SVM approach, suggesting the effectiveness of deep learning techniques in capturing complex facial features.
4. Matching Techniques: Matching techniques included cosine similarity, k-means clustering, and k-nearest neighbors. These methods allowed for the comparison of facial features and the grouping of similar faces. It was particularly interesting that the KNN model with a parameter of K=1 did not cause overfitting, which is the general rule of thumb. This is likely due to the great quality of the embeddings generated by Facenet and fed to the KNN model.
5. GCPD/LFW Datasets: The K-means model fitted on the GCPD dataset had a silhouette score near 0, which indicates that the distance between the clusters is not significant. This implies that the model can be made much stronger. This result is expected since no preprocessing step of the GCPD Dataset was performed to handle issues of alignment and occlusion unlike LFW.
6. MTCNN: It was observed that some objects were incorrectly detected as faces by the SOTA detector. This contributed to our training set including a few outliers and thus affecting the models' performances.

Drawbacks:

1. Because our implemented HOG is designed with a lot of nested for-loops and therefore takes a lot of time to return detected faces, MTCNN was preferred for quick turnaround for subsequent sections of the clustering pipeline. Given more time to work on the project we would optimize on the computation speed by tuning some of the design parameters of the HOG like the number of grid cells and the size of the normalization block.
2. Performance Discrepancy between CNN and SVM: A significant drawback was observed as the CNN outperformed the SVM in face recognition tasks. The analysis needs to delve into the reasons behind this performance gap.
3. Potential Causes: The limitations of SVM in handling complex and non-linear relationships in facial features may contribute to its poor performance along with the structure of the LFW dataset as mentioned prior. Convolutional Neural Networks, on the other hand, are designed to automatically learn hierarchical representations, allowing them to capture intricate patterns and dependencies in the data.
4. Feature Representation Complexity: Facial features are inherently complex, and the fixed feature extraction strategy of SVM may not be sufficiently adaptive. CNNs excel in learning hierarchical features, enabling them to better represent the intricate nuances present in facial images.
5. Data Size and Diversity: The performance discrepancy may also stem from the size and diversity of the dataset. CNNs often require larger and more diverse datasets to generalize well, while SVM might struggle with the complexity of facial feature representations in smaller datasets.

Potential Solutions:

1. Experiment with Different SVM Configurations: Fine-tune SVM parameters, explore different kernels, and experiment with variations to enhance its adaptability to complex facial features.
2. Ensemble Approaches: Combine the strengths of SVM and CNN through ensemble techniques to leverage the complementary aspects of both methods.
3. Data Augmentation and Transfer Learning: Augment the dataset and employ transfer learning techniques to enhance the generalization ability of both SVM and CNN models.
4. Investigate Hybrid Models: Explore hybrid models that integrate aspects of both traditional machine learning and deep learning to harness the advantages of each approach.

While all methods showed promise, the standout success of CNNs over SVM in face recognition indicates the potency of deep learning for capturing intricate facial features. The positive outcomes from clustering techniques and the project's commitment to continuous improvement through ensemble methods and hybrid models provide a solid foundation for future advancements in automated face recognition and image grouping. The project's success signals a bright future for the integration of deep learning in image analysis and pattern recognition.

5 STATEMENT OF INDIVIDUAL CONTRIBUTION

Veda Kailasam: In the pursuit of advancing facial feature extraction and prediction models, my contributions to the project have been multifaceted and instrumental. Firstly, I explored the application of Principal Component Analysis (PCA) for facial feature extraction, leveraging its capability to reduce dimensionality while retaining essential

information. Subsequently, I implemented a Support Vector Machine (SVM) for prediction, utilizing the features extracted by PCA. I embarked on the implementation of a pre-trained InceptionV2Resnet model to extract facial embeddings. However, the performance on the Labeled Faces in the Wild (LFW) dataset did not meet expectations when compared to a convolutional neural network (CNN) model that I independently trained. This divergence prompted me to delve into the nuances of model architecture and training methodologies. I initiated the training of a CNN model from scratch, meticulously iterating through 60 epochs. The results were commendable, achieving an accuracy of 93.7% on the LFW dataset. This underscores the importance of tailoring the model architecture to the specific requirements of facial feature extraction and prediction tasks. I implemented a face matching system using cosine similarity for comparing face embeddings. Face embeddings, derived from the CNN model trained from scratch, serve as compact and meaningful representations of facial features. The cosine similarity metric was employed as a measure of similarity between these embeddings. The utilization of cosine similarity in this context demonstrated the project's commitment to employing appropriate and effective similarity metrics for the task at hand. I explored KMeans clustering on the embeddings extracted from a pre-trained FaceNet model. This endeavor provided insights into the inherent structure of the data and its clustering patterns. To visualize the results, I generated t-SNE plots, offering a comprehensive visual representation of the facial feature embeddings in a reduced-dimensional space. Collectively, my contributions encompassed a thorough exploration of dimensionality reduction techniques, model architectures, and clustering methodologies. The amalgamation of these efforts culminated in the development of a robust CNN model, showcasing a significant leap in accuracy on the LFW dataset. Additionally, the exploration of clustering techniques provided valuable insights into the distribution and grouping of facial feature embeddings. Through these endeavors, I aimed to push the boundaries of facial feature extraction and prediction, contributing to the project's overarching goal of advancing the field.

Jorge Chavez: My endeavors were focused on refinement and tuning of the PCA and SVM algorithms, exploring parallel approaches for speeding up the process, the incorporation of MTCNN, and creating a basic framework using Angular for the front end application, and Flask as the backend server. Through my work, we were able to discover and discuss the results of classic machine learning algorithms and probe their effectiveness on a dataset such as LFW. By incorporating MTCCN, we were able to refine feature extraction and recognition techniques before our final deployment of the HOG facial extractor. Finally, with a basic full stack framework, we were able to demonstrate these results in a more modern UI which we hope to later deploy on a web service such as AWS.

Paul Osuma: My contributions on the project covered face detection and feature matching using SVM on PCA generated embeddings. For face detection, I sought to apply rudimentary computer vision techniques to build a face detector from scratch. My efforts in executing this stretched from the basics of detecting edges using edge filters to training a machine learning model using SVM. I was happy to see the HOG model not only achieve a high accuracy on the training but also perform well in drawing correct bounding boxes around faces of different scale, occluded faces and non-front facing faces. I worked with Jorge Chavez on clustering images encoded with PCA using SVM. In Particular, I worked on trying out different training variations from SVM from linearSVC to general SVC and also plotting the results achieved for the dataset with 60 or more images per person. I was glad to learn through debugging our code why datasets with less images per person gave us poor results and how to work with GPUs on google Colab for faster processing speeds.

Philmon Roberts:
I gathered the GCPD datasets so that we can train a model on a dataset closely related to photo albums. I trained a KNN and SVM on the entire LFW dataset, achieving accuracies greater than 84%. In addition, I implemented the clustering algorithm using K-means and demonstrated it's performance in the video.

6 REFERENCES

A. https://learnopencv.com/histogram-of-oriented-gradients/
B. https://jakevdp.github.io/PythonDataScienceHandbook/05.14-image-features.html
C. https://www.researchgate.net/figure/Example-of-occluded-face-images_fig1_347538199
D. https://www.freepik.com/free-photos-vectors/black-man-face
E. https://www.shutterstock.com/image-photo/portrait-thoughtful-handsome-man-looking-away-287094836
F. https://www.tandfonline.com/doi/epdf/10.1080/21642583.2020.1836526?needAccess=true
G. https://www.kaggle.com/code/serkanpeldek/face-recognition-on-olivetti-dataset/notebook

7 REFERENCES

1. Google Drive