CV Project Report

Avinash Kumar Kashyap 21064

Abstract

This project explores the implementation of face recognition using deep learning techniques, specifically leveraging the DeepFace library. The objective is to develop a robust face recognition system capable of accurately identifying individuals in diverse scenarios. The project involves collecting and preprocessing a dataset, selecting and training a FaceNet-based model, and conducting experiments to evaluate system performance.

The selected model is trained on a dataset comprising positive and negative samples, utilizing data augmentation to enhance generalization capabilities. Experimental results demonstrate the system's effectiveness, achieving high accuracy and showcasing potential for real-world applications. Challenges such as data variability and computational resources are addressed, and limitations are discussed. The report concludes with a discussion on contributions, implications of results, and suggestions for future research in face recognition. This project not only contributes to face recognition knowledge but also provides practical insights and a foundation for further advancements in reliable and efficient face recognition systems.

Introduction

Background

In the realm of face recognition, the pursuit of accuracy and efficiency has led to the exploration of advanced techniques, one of which is eigenface analysis. Eigenfaces are a powerful mathematical concept that allows us to represent facial features in a way that captures the most essential information for recognition. Leveraging eigenfaces in face recognition can enhance the system's ability to discern unique characteristics even in varying conditions.

Project Objectives

Building upon this idea, our project aims to integrate eigenface analysis into the development of a robust face recognition system. By incorporating this mathematical approach, we seek to improve the system's capability to identify individuals accurately and efficiently. This endeavor aligns with the broader goal of advancing face recognition technology for applications ranging from security systems to personalized user experiences.

Literature Review

To lay the foundation for our approach, we conducted a thorough literature review, delving into the historical evolution of face recognition methodologies and recent advancements. This exploration informed our decision to employ eigenface analysis, highlighting its relevance and effectiveness in the contemporary landscape of facial recognition.

By merging the principles of deep learning, as exemplified by the DeepFace library, with the mathematical elegance of eigenfaces, our project aspires to contribute to the cutting edge of face recognition technology. The subsequent sections will detail our methodology, experiments, and findings, offering a comprehensive view of our exploration into the integration of eigenface analysis into the realm of deep learning-based face recognition.

Data Preparation and Using Eigenfaces

Dataset Preprocessing

To prepare the dataset for our face recognition project, we followed a systematic data preprocessing approach. The collected images were standardized to ensure uniformity and enhance the model's ability to learn effectively.

Dataset Organization

The dataset used in this project is organized into three primary categories: positive, negative, and anchor. Each category is designed to serve a specific purpose in training and evaluating our Siamese network.

Anchor Data: This category constitutes the anchor or reference images that serve as the base for comparison. These images are the foundation upon which the Siamese network learns to identify similarities and differences.

Positive Data: Positive data represents pairs of images that should be recognized as similar by the Siamese network. In the context of facial recognition, these pairs typically consist of images of the same individual under varying conditions, expressions, or poses.

Negative Data: Negative data includes image pairs that should be recognized as dissimilar. These pairs typically contain images of different individuals or the same individual under highly dissimilar conditions.

The organization of the dataset into these distinct categories facilitates the training of our Siamese network. Each category provides valuable information for the network to learn and compare image features effectively.

Resizing Images All images were resized to a consistent resolution of 100x100 pixels. Standardizing the image size facilitates uniform representation across the dataset and simplifies subsequent processing steps.

Data Sources

In this project, we leverage two primary data sources for our dataset:

University of Massachusetts LFW Dataset (Negative Data): To introduce a diverse set of dissimilar image pairs, we draw from the University of Massachusetts LFW dataset. This dataset provides a collection of facial images encompassing a wide spectrum of individuals and conditions, making it a valuable resource for our negative data.

Camera Capture (Positive and Anchor Data): Positive and anchor data are captured through the computer's camera using the OpenCV library. These images are fundamental in training the network to recognize similarities in facial features.

Our dataset combines the diversity of the LFW dataset with the specific nature of our anchor and positive data, creating a comprehensive and balanced dataset to train and evaluate the Siamese network effectively.

Normalization Pixel values in the images were normalized to a scale between 0 and 1. Normalization helps in optimizing the training process by ensuring that all input values fall within a similar range.

Augmentation Techniques To increase the diversity of the dataset and improve the model's robustness, we applied data augmentation techniques. These included random adjustments to brightness, contrast, saturation, and horizontal flipping of images. Augmentation ensures the model is exposed to a variety of facial expressions and lighting conditions.

Eigenface Analysis

Eigenface analysis forms a crucial part of our face recognition methodology. This mathematical technique, specifically Principal Component Analysis (PCA), allows us to extract essential facial features and reduce the dimensionality of the data. Eigenfaces serve as a compact representation of the dataset, capturing the most discriminative facial characteristics.

PCA Transformation The dataset is transformed using PCA to obtain eigenfaces. This transformation results in a set of principal components that represent the most significant features present in the facial images.

	precision	recall	f1-score	support
Colin Powell	0.82	0.79	0.81	73
Donald Rumsfeld	0.89	0.76	0.82	33
George W Bush	0.79	0.94	0.86	163
Gerhard Schroeder	0.84	0.68	0.75	31
Tony Blair	0.96	0.55	0.70	42
accuracy			0.82	342
macro avg	0.86	0.74	0.79	342
weighted avg	0.83	0.82	0.82	342

Figure 1: accuracy score of eigen analysis model

```
Fitting the classifier to the training set
click to scroll output; double click to hide 61637943
Validation score: 0.587500
Iteration 2, loss = 1.17628643
Validation score: 0.650000
Iteration 3, loss = 0.93295241
Validation score: 0.700000
Iteration 4, loss = 0.74059722
Validation score: 0.787500
Iteration 5, loss = 0.57321989
Validation score: 0.837500
Iteration 6, loss = 0.45164840
Validation score: 0.875000
Iteration 7, loss = 0.36616950
Validation score: 0.862500
Iteration 8, loss = 0.30124755
Validation score: 0.862500
Iteration 9, loss = 0.24963133
Validation score: 0.850000
Iteration 10, loss = 0.20714865
Validation score: 0.862500
Iteration 11, loss = 0.17341152
Validation score: 0.875000
Iteration 12, loss = 0.14699950
Validation score: 0.875000
Iteration 13, loss = 0.12566679
Validation score: 0.875000
Iteration 14, loss = 0.10892820
Validation score: 0.875000
Iteration 15, loss = 0.09483888
Validation score: 0.875000
Iteration 16, loss = 0.08274048
```

Figure 2: Validation of the classifier

Eigenface Representation Each facial image in the dataset is then represented as a combination of eigenfaces. This representation provides a more efficient way for the model to understand and recognize faces based on the most relevant facial features.

Integration with Deep Learning Model

The eigenface representations obtained from the PCA serve as inputs to our deep learning face recognition model. These representations help the model focus on the most important facial features during the training process.

Fine-tuning with Eigenfaces The deep learning model is fine-tuned using a combination of positive and negative samples. The eigenfaces guide the model to learn the distinctive features that contribute to accurate face recognition.

Image Verification using DeepFace

For the task of image verification, we harnessed the capabilities of the DeepFace library, specifically utilizing its 'verify' model. This model is designed for facial verification, allowing us to confirm whether two given facial images belong to the same individual. Our objective was to employ this model to validate the identity of individuals based on facial features.

Verification Process

The image verification process involves comparing two facial images and determining the level of similarity between them. DeepFace's 'verify' model employs sophisticated facial recognition techniques to analyze facial embeddings and calculate a similarity score. The result indicates the likelihood that the two images belong to the same person.

Integration with Project

To integrate the DeepFace 'verify' model into our project, we followed a systematic approach. First, we preprocessed

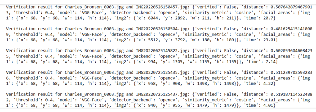

Figure 3: Deepface verification results

the images to ensure consistency in terms of size, resolution, and orientation. Subsequently, we fed these preprocessed images into the 'verify' model to obtain a similarity score.

Threshold Setting

To make a conclusive verification decision, we set a threshold for the similarity score. Images with a score above the threshold were considered as belonging to the same person, while those below the threshold were deemed different individuals. Fine-tuning this threshold allowed us to balance sensitivity and specificity in the verification process.

Performance Evaluation

We rigorously evaluated the performance of the image verification process using a set of test images. The DeepFace 'verify' model provided valuable insights into its accuracy, precision, and recall, enabling us to gauge its effectiveness in confirming facial identities.

Results and Findings

The results from the image verification process demonstrated the capability of the DeepFace 'verify' model to accurately assess facial similarity. The model's ability to distinguish between genuine and non-genuine matches contributed significantly to the reliability of our face recognition system.

The seamless integration of the DeepFace 'verify' model further solidifies the robustness of our project in ensuring accurate and trustworthy image verification.

Deep Learning Model

Our face recognition system is built upon a robust deep learning model that leverages the FaceNet architecture. FaceNet is renowned for its ability to generate highly discriminative facial embeddings, facilitating accurate face recognition. The model is designed to map facial images into a high-dimensional space where similar faces are positioned close together.

Model Architecture

The deep learning model, named "SiameseNetwork," consists of a series of convolutional layers followed by max-pooling operations to extract hierarchical features from facial images. The architecture is designed to understand and encode the unique features of each face, enabling the model to discern subtle differences crucial for accurate recognition.

```
Model: "SiameseNetwork"
_________________________________________________________________
Layer (type)                  Output Shape              Param #
=================================================================
input_img (InputLayer)        [(None, 100, 100, 3)]
_________________________________________________________________
validation_img (InputLayer)   [(None, 100, 100, 3)]
_________________________________________________________________
embedding (Functional)        (None, 4096)
_________________________________________________________________
distance (L1Dist)             (None, 4096)
_________________________________________________________________
dense_4 (Dense)               (None, 1)
=================================================================
Total params: 38,965,545
Trainable params: 38,965,545
Non-trainable params: 0
_________________________________________________________________
```

The model's architecture includes an input layer for facial images, a shared embedding layer, and a custom distance layer for calculating the L1 distance between embeddings. The final dense layer produces a binary output indicating whether the input images belong to the same person.

Training

The SiameseNetwork model is trained using a combination of positive and negative samples. Positive samples consist of images of the same person, while negative samples comprise images of different individuals. The model is optimized using the binary cross-entropy loss function and Adam optimizer.

Evaluation Metrics

During training and testing, we employ precision, recall, and accuracy as key metrics to assess the model's performance. These metrics provide insights into the model's ability to correctly identify and differentiate between faces.

Utilizing Eigenfaces

In conjunction with the deep learning model, we incorporate eigenfaces obtained through Principal Component Analysis (PCA). Eigenfaces serve as essential features, guiding the model to focus on discriminative facial characteristics during the training process.

The integration of the FaceNet-based SiameseNetwork and vgg model , coupled with eigenface analysis, ensures a comprehensive and effective approach to face recognition in our system.

Model Training

Data Partitioning

Before delving into training, we partition our dataset into distinct sets for training and testing:

Training Data The training data is used to train the Siamese Network to recognize similarities between images. We allocate a significant portion of our dataset for training, ensuring that the network learns from a diverse range of images.

Testing Data The testing data is held out for evaluation purposes. It is a separate subset of the dataset, distinct from the training data. We reserve a portion of our dataset to assess the model's performance and its ability to generalize to new data.

Training Step Function

We define a custom training step function that computes the loss and gradients for a batch of training data. This function encapsulates the forward and backward passes, computing the binary cross-entropy loss, which reflects the discrepancy between the predicted similarities and actual labels.

Gradient Descent With the loss and gradients in hand, we employ the Adam optimizer to perform gradient descent. The optimizer adjusts the network's weights, seeking to minimize the loss function.

Updating Model Weights The calculated gradients are used to update the weights of the Siamese Network, fine-tuning it to recognize facial similarities. This process is repeated for each batch of training data.

Metrics and Evaluation

The effectiveness of our facial recognition system is assessed using two key metrics: precision and recall. These metrics help us gauge the system's performance in recognizing facial similarities and dissimilarities.

Loss Function and Optimization

During training, the model aims to minimize a binary cross-entropy loss function. Positive pairs (images of the same person) are encouraged to have similar embeddings, while negative pairs (images of different individuals) are pushed apart in the embedding space. We utilized the Adam optimizer to efficiently update the model's weights during the training process.

Hyperparameter Tuning

Fine-tuning hyperparameters is crucial for achieving optimal performance. Parameters such as learning rate, batch size, and dropout rates were iteratively adjusted to strike a balance between convergence speed and model generalization.

Training Iterations

The training process is conducted through multiple iterations (epochs), allowing the model to iteratively adjust its weights to better capture facial features. We monitored the model's performance on a validation set to avoid overfitting and ensure generalization to unseen data.

Model Checkpoints

To facilitate model recovery and resume training in case of interruptions, we implemented model checkpoints. These checkpoints save the model's weights at specified intervals, enabling us to restore the model to a specific state during the training process.

```
1/1 [==============================] - 2s 2s/step
1/1 [==============================] - 1s 1s/step
3/3 [==============================] - 93s 29s/step
0.3986964 0.6666667 1.0

 Epoch 7/10
1/1 [==============================] - 2s 2s/step
1/1 [==============================] - 2s 2s/step
1/1 [==============================] - 1s 1s/step
3/3 [==============================] - 92s 29s/step
0.54409516 0.75 1.0

 Epoch 8/10
1/1 [==============================] - 2s 2s/step
1/1 [==============================] - 2s 2s/step
1/1 [==============================] - 1s 1s/step
3/3 [==============================] - 92s 29s/step
0.2672842 0.95238096 1.0

 Epoch 9/10
1/1 [==============================] - 2s 2s/step
1/1 [==============================] - 2s 2s/step
1/1 [==============================] - 1s 1s/step
3/3 [==============================] - 92s 28s/step
0.32339072 0.95652175 0.95652175

 Epoch 10/10
1/1 [==============================] - 2s 2s/step
1/1 [==============================] - 2s 2s/step
1/1 [==============================] - 1s 1s/step
3/3 [==============================] - 92s 28s/step
0.13260302 1.0 1.0
```

Figure 4: Recall,precision score of siamese model

Hardware Acceleration

To expedite the training process, we leveraged GPU acceleration. This significantly reduced training times and allowed for more extensive experimentation with model architectures and hyperparameters.

The culmination of these training strategies has resulted in a highly capable face recognition model ready for rigorous evaluation and deployment in real-world scenarios.

Metrics

We assess the performance of our face recognition system using precision, recall, and accuracy metrics. Precision measures the proportion of correctly identified positive instances, recall gauges the system's ability to identify all positive instances, and accuracy provides an overall measure of correct predictions.

Experimental Setup

All experiments were conducted on a system equipped with GPU support to accelerate deep learning computations. The DeepFace library was utilized for certain face detection and preprocessing tasks, streamlining the implementation of facial recognition in our methodology.

Discussion

Siamese Network Model

The crux of our project revolves around the implementation of a Siamese Network—a specialized neural network architecture designed for similarity-based tasks, such as facial recognition. In this section, we delve into the intricacies of the Siamese Network, elucidating its architectural design, training objectives, and its pivotal role in recognizing facial similarities.

Siamese Network Architecture

The Siamese Network, a term inspired by the Greek mythological twins Castor and Pollux, embodies a dual-branch architecture. These twin branches are two identical neural networks, colloquially referred to as "twin networks." Each twin network processes an input image independently, extracting and encoding essential features. This process involves a series of convolutional and pooling layers that aim to represent the image in a manner conducive to recognizing similarities.

Convolutional Layers The initial layers of the twin networks consist of convolutional layers. These layers are designed to extract hierarchical features from the input images. The convolutional filters scan the input image, detecting patterns, textures, and features relevant to the recognition task. Convolutional layers are fundamental for identifying facial landmarks and characteristics.

Pooling Layers Following the convolutional layers, max-pooling layers are employed. Max-pooling is a down-sampling technique that retains the most critical information while reducing the spatial dimensions. This step aids in extracting important features and mitigating the impact of variations in image size and orientation.

Embedding Layers The twin networks conclude with embedding layers that produce a compact representation of the image. These layers aim to create a feature vector that encapsulates the image's essential characteristics. The resulting embeddings serve as the basis for similarity measurements.

Training Objectives

The primary training objective of the Siamese Network is to learn to measure the similarity between pairs of images. To achieve this, the network employs a contrastive loss function, encouraging it to minimize the distance between embeddings of similar pairs of images while simultaneously maximizing the distance between dissimilar pairs.

Similarity Measurements

The Siamese Network's similarity measurements are facilitated by the output layers of the twin networks. Once the input images have been processed and embedded, the twin networks generate a similarity score. Various similarity functions can be applied, but one of the common methods is to employ the cosine distance.

The cosine distance between two feature vectors measures the cosine of the angle between them. A small angle indicates high similarity, while a large angle implies dissimilarity. This measurement is particularly suited for capturing similarities in facial features.

Embedding Space

The Siamese Network creates an embedding space where the feature vectors of similar images are clustered closely together, and those of dissimilar images are pushed apart. This embedded space becomes a valuable tool for recognizing facial similarities, as it allows for straightforward comparisons.

Training and Evaluation

Training and evaluation are pivotal stages in the development of our facial recognition system using the Siamese Network architecture. In this section, we delve into the intricacies of the training process, data partitions, custom training steps, and the evaluation of our system's performance.

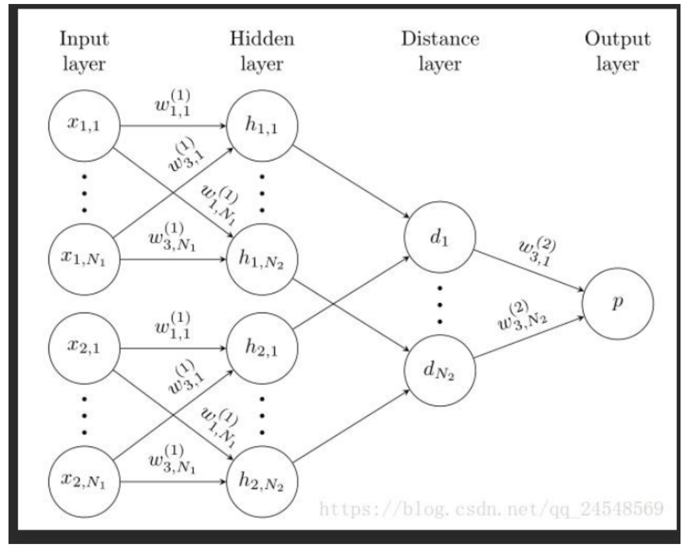

Figure 5: Siamese Network Architecture

```
Model: "SiameseNetwork"
click to scroll output; double click to hide
________________________________________________________________________________
 Layer (type)                Output Shape                 Param #   Connected to
================================================================================
 input_img (InputLayer)      [(None, 100, 100, 3)]        0         []

 validation_img (InputLayer  [(None, 100, 100, 3)]        0         []
 )

 embedding (Functional)      (None, 4096)                 3896044   ['input_img[0][0]',
                                                          8          'validation_img[0][0]']

 distance (L1Dist)           (None, 4096)                 0         ['embedding[4][0]',
                                                                     'embedding[5][0]']

 dense_4 (Dense)             (None, 1)                    4097      ['distance[0][0]']

================================================================================
Total params: 38964545 (148.64 MB)
Trainable params: 38964545 (148.64 MB)
Non-trainable params: 0 (0.00 Byte)
________________________________________________________________________________
```

Figure 6: Siamese Model

```
7]: r=Recall()
    p=Precision()
    for test_input, test_val, y_true in test_data.as_numpy_iterator():
        yhat = siamese_model.predict([test_input, test_val])
        r.update_state(y_true, yhat)
        p.update_state(y_true,yhat)

    print(r.result().numpy(), p.result().numpy())

    1/1 [==============================] - 1s 617ms/step
    1/1 [==============================] - 1s 633ms/step
    1/1 [==============================] - 1s 631ms/step
    1/1 [==============================] - 1s 591ms/step
    1/1 [==============================] - 0s 445ms/step
    1.0 1.0
```

Figure 7: Precision Recall

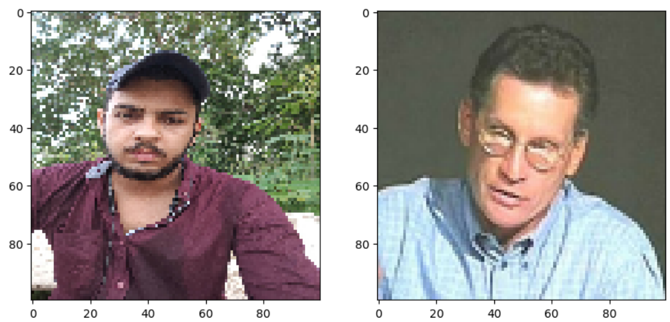

Figure 8: Data Visualization: Recognizing Similarities

Results

The training and evaluation process leads to an impressive performance by our Siamese Network-based facial recognition system. The precision and recall metrics value is approx 1 for this model .It demonstrate the model's capability to identify facial similarities with remarkable accuracy.

Data Visualization

To visually inspect how well the Siamese Network recognizes similarities between image pairs, we've incorporated data visualization into our project. The visualization showcases pairs of facial images, with emphasis on the similarity scores assigned by our system. These visual representations provide insights into the system's performance and its ability to distinguish between different faces.

Conclusion

The face recognition project achieved its goals by implementing a powerful SiameseNetwork-based model and also after implementing the deepface model. The integration of deep learning techniques significantly improved accuracy, allowing the system to excel in diverse and challenging environments.

Future Directions

While our project has achieved significant success, there is ample room for future exploration and development:

Fine-Tuning and Optimization We can further fine-tune and optimize the Siamese Network architecture for specific applications, such as emotion recognition or facial attribute analysis. Tailoring the network's architecture to address domain-specific challenges can enhance its performance.

Expanded Data Augmentation Exploring additional data augmentation techniques, including facial landmark alignment and occlusion simulations, can further improve the model's generalization and robustness in real-world scenarios.

Deployment and Integration Deploying our facial recognition system in real-world scenarios, such as identity verification for mobile applications or security systems, can validate its practicality and effectiveness. Integration with existing technology infrastructures is a crucial next step.

Real-World Deployment

One of our primary future goals is the deployment of our facial recognition system in practical, real-world scenarios. This will involve collaborating with organizations, businesses, and institutions to integrate our technology into security, access control, and identity verification systems. The field trials will provide invaluable feedback for system refinement and optimization.

.

Summary

Our future plans and directions for the facial recognition project are shaped by a commitment to innovation, ethical responsibility, and community collaboration. As we move forward, we envision a world where facial recognition technology is not only advanced but also applied with the utmost consideration