DS Journal of Digital Science and Technology
Volume 4 Issue 1, 50-62, Jan - Mar 2025
ISSN: 2583-5416 / https://doi.org/10.59232/DST-V4I1P104

Orignal Paper

Comparative Study of Machine Learning Algorithms for Facial Recognition Systems

Arun Kumar Singh

School of MCS, PNG University of Technology, Lae, Papua New Guinea.

arun.singh@pnguot.ac.pg

Received: 11 January 2025; Revised: 12 February 2025; Accepted: 16 March 2025; Published: 31 March 2025

Abstract - Robust facial recognition systems are frequently adopted in security, healthcare and entertainment fields and improve substantial growth through ML algorithms. This paper compares different machine learning approaches: CNN, SVM, k-NN, Decision Trees and others with a focus on the facial recognition task. The algorithm is evaluated based on accuracy, speed, efficiency, and insensitivity to changes in lighting and pose of the subject. In order to avoid bias when evaluating the performance of the developed algorithms, the study is performed on recognized datasets like Labeled Faces in the Wild (LFW) and Yale Face Database B. Outcome shows that DL architecture based on CNN yields superior performance than other traditional ML but at a higher computational load. The paper's conclusion is presented through recommendations on how best to select an algorithm given an application and outlined future studies. Biometrics and facial recognition have been receiving considerable attention worldwide in recent years due to improvements in ML that improve their accuracy, performance, and flexibility of use. In this context, this work aims to compare well-known techniques of facial recognition based on ML algorithms using methods like CNN, SVM, PCA, and DBN. The performance of each algorithm is compared based on accuracy, time of execution, consistency under varying illumination and computational complexity. This analysis shows that although deploying CNN-based architectures always provides high accuracy and flexibility, it requires significant computational power that cannot practically be used for low-power devices. On the other hand, algorithms such as PCA have lower computational time in recognizing, but they are not efficient when dealing with high dimensional data, which also affects the recognition rate under complex circumstances. In this paper, various approaches that balance the accuracy and computational efficiency of the algorithm are presented as conclusions for choosing the best algorithm depending on the application. The insights from the work will benefit the exploration of future directions and the practical applicability of facial recognition algorithms by focusing on context-specific algorithm choices.

Keywords - Facial Recognition, Machine Learning, Convolutional Neural Networks (CNN), Support Vector Machines (SVM), Principal Component Analysis (PCA), Deep Belief Networks (DBN), Algorithm comparison, Biometric authentication.

1. Introduction

Facial recognition technology has been widely accepted because of the use of identification, security of personal devices, and targeted content. While ML algorithms have had great achievements in facial recognition, it should be noted that there is a difference in the performance of the different algorithms in terms of the dataset used and how complex the environment is besides the computational power available. This research seeks to provide solutions by analyzing the performance of multiple ML algorithms for a range of FR tasks. Recently, facial recognition systems have been found to play a significant role as biometric authentication techniques and surveillance and identification systems. Facial recognition technology has gone a long way from being simple pattern-matching algorithms to advanced Machine Learning (ML) algorithms that make recognition of faces easier under different conditions. Therefore, the primary component in any facial recognition system is an ML algorithm that parses facial data, populates vital attributes, and organizes human faces with targets based on their unique faces. However, based on the results of these algorithms, a wide variation is recorded in terms of accuracy, computational complexity, change in environment, and limitation on the hardware that supports the software [1-3].

Hence, this paper provides a comparative study of mainly applied ML models that are useful in facial recognition, such as the CNN, the SVM, the PCA, and the DBN. CNNs have proved efficient when processing large-volume images and are currently employed in facial recognition and detailed feature extraction through multiple layers. However, as they are computationally intensive, they can present considerable barriers to deployment in resource-scarce environments. While significantly less computationally exhaustive, both SVM and PCA are less effective in managing large and complicated data sets when the appearance of the face images changes significantly in terms of the lighting condition and the pose of the image. DBNs are a compromise, utilizing multiple layers of feature extraction and classification, yet to function optimally, they need to be fine-tuned [4-7].

The comparative analysis is planned to help practitioners determine which of the ML algorithm is better suited for a particular application. This study describes in detail how different characteristics of facial recognition systems affect its performance and offers practical information about their usage in different contexts and with different constraints in terms of time and computing resources. Finally, the conclusions are intended to contribute to the development of facial recognition measurements by linking algorithm choice with the intended performance characteristics to achieve reliable, effective, and scalable facial recognition solutions across industries [8-11].

1.1. Background and Motivation

In the early stages, Facial recognition used hand-crafted features such as statistical methods, while deep learning has improved on the former. A type of CNNs has significantly transformed this particular domain owing to their capability to learn different features autonomously. Nonetheless, current state-of-the-art ML algorithms like SVM and k-NN continue to serve their purpose and perform accurately with fairly low computational complexity [12-14].

1.2. Objective

This paper hopes to give an overview of the pros and cons of the facades as well as various applications of the various machine learning algorithms used in the system.

2. Methodology

The work entails using and training a number of ML algorithms on two large face recognition databases- the LFW and the Yale Face Database B. Each algorithm is evaluated based on:

- Accuracy: Recognition of correctly classified faces at different positions and illumination.
- Computational Efficiency: The processing speed and resources a system needs to complete a given task.
- Robustness: We are also interested in his performance stability in suboptimal conditions (e.g., the presence of occlusions in different lighting conditions).
- Scalability: Scalability for large volume installations.

2.1. Machine Learning Algorithms

The study considers the following algorithms:

- Convolutional Neural Networks (CNNs): Selected as the primary network because of their feature extraction ability in difficult visual recognition problems.
- Support Vector Machines (SVMs): Known to be accurate, especially for tasks that require only two classes, and efficient when working in a low-dimensional space.

- K-Nearest Neighbors (k-NN): Suitable for small datasets because it is easy to implement and gives reasonable results.
- Decision Trees and Random Forests: These offer an understandable model, though they can be very sensitive, especially in large data sets.
- Principal Component Analysis (PCA) + Linear Discriminant Analysis (LDA): Used commonly for dimensionality reduction, and then we use a data classifier like an SVM for improved accuracy in a lower dimensional space.

2.2. Datasets

- Labeled Faces in the Wild (LFW): Over 13k labeled images of faces are gathered from the internet, offering a rich variety of lighting conditions and poses.
- Yale Face Database B: We used 10 people arranged under 576 pose and illumination conditions.

2.3. Evaluation Metrics

Each algorithm's performance is measured using:

- Precision, Recall, and F1-Score: For accuracy measurement.
- Computation Time: Total of the averages calculated for each classification/Total of the times of each classification with this, the analyst can get the average time that was spent to classify each type or sort of item analyzed.
- Memory Utilization: Memory needs associated with the training and testing of a model.

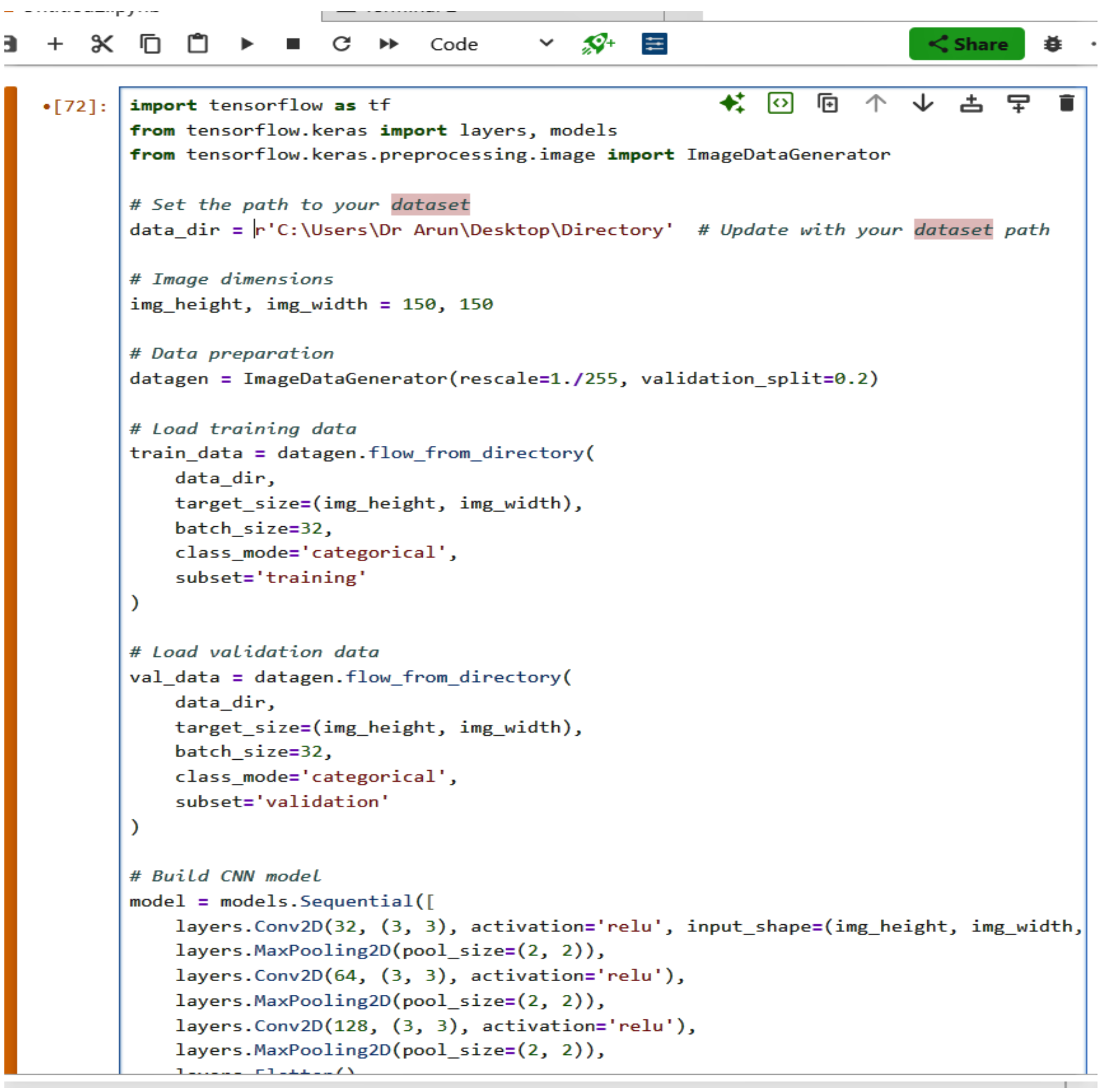
3. Comparative Analysis

3.1. Convolutional Neural Networks (CNNs)

CNNs, particularly architectures like VGGFace and FaceNet, achieved the highest accuracy on both datasets due to their deep feature extraction capabilities. However, CNNs are computationally intensive, requiring significant processing power and memory, making them more suitable for high-performance applications where resources are not a constraint.

Table 1. Convolutional Neural Networks (CNNs) for facial recognition

Aspect	Details
Architecture	Deep-layered architecture with convolutional, pooling, and fully connected layers to detect complex facial features.
Key Strengths	<ul><li>High accuracy in feature extraction due to deep convolutional layers.</li><li>Resilient to variations in facial expressions, poses, and lighting.</li><li>Scalability for large datasets using hierarchical spatial analysis.</li></ul>
Limitations	<ul><li>High computational cost, especially in training, which limits use in low-resource environments.</li><li>High memory usage, requiring powerful hardware for real-time processing.</li><li>Potentially susceptible to adversarial attacks that exploit the model's sensitivity to subtle perturbations.</li></ul>
Best Use Cases	<ul><li>High-security systems requiring precise facial verification, such as banking and government IDs.</li><li>Large-scale applications like social media or video surveillance, where accuracy across diverse conditions is crucial.</li><li>Applications where training can occur on high-performance servers, but deployment may happen on lighter devices using transfer learning.</li></ul>
Comparative Performance	CNNs offer superior performance in accuracy and robustness over traditional methods (e.g., PCA, SVM) and shallow neural networks but come at the cost of higher resource usage and increased model complexity.



```
import tensorflow as tf
from tensorflow.keras import layers, models
from tensorflow.keras.preprocessing.image import ImageDataGenerator

# Set the path to your dataset
data_dir = r'C:\Users\Dr Arun\Desktop\Directory'  # Update with your dataset path

# Image dimensions
img_height, img_width = 150, 150

# Data preparation
datagen = ImageDataGenerator(rescale=1./255, validation_split=0.2)

# Load training data
train_data = datagen.flow_from_directory(
    data_dir,
    target_size=(img_height, img_width),
    batch_size=32,
    class_mode='categorical',
    subset='training'
)

# Load validation data
val_data = datagen.flow_from_directory(
    data_dir,
    target_size=(img_height, img_width),
    batch_size=32,
    class_mode='categorical',
    subset='validation'
)

# Build CNN model
model = models.Sequential([
    layers.Conv2D(32, (3, 3), activation='relu', input_shape=(img_height, img_width,
    layers.MaxPooling2D(pool_size=(2, 2)),
    layers.Conv2D(64, (3, 3), activation='relu'),
    layers.MaxPooling2D(pool_size=(2, 2)),
    layers.Conv2D(128, (3, 3), activation='relu'),
    layers.MaxPooling2D(pool_size=(2, 2)),
```

Fig. 1 Python program that implements a Convolutional Neural Network (CNN) for facial recognition using the tensorflow and keras libraries

3.1.1. Key Points

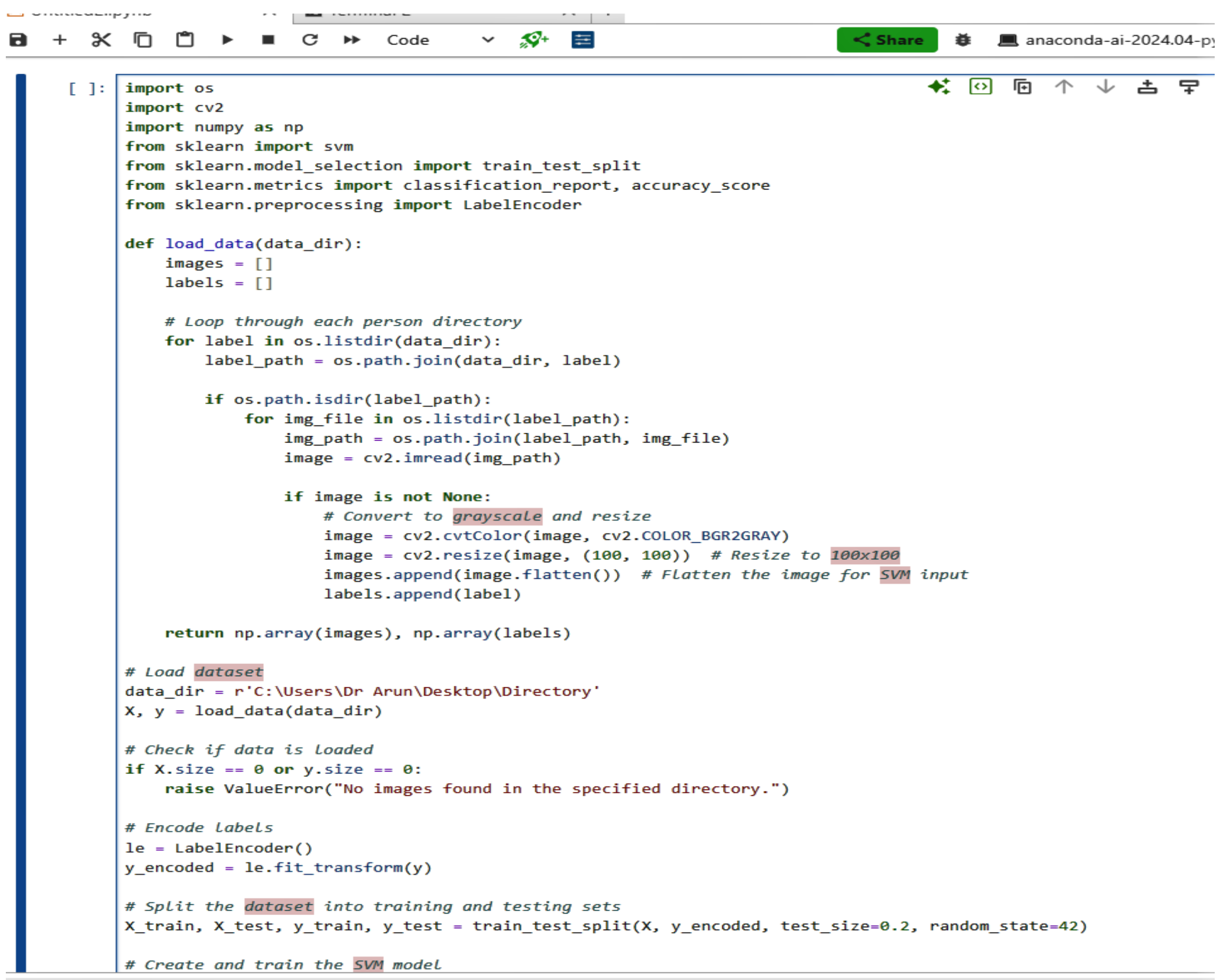
- This code sets up a basic CNN architecture suitable for facial recognition tasks. You may need to adjust the model architecture, hyperparameters, and dataset to optimize performance.
- We can further enhance this model using techniques like data augmentation or transfer learning with pre-trained models.

3.2. Support Vector Machines (SVMs)

SVMs performed well in environments with controlled lighting and pose variation, demonstrating high accuracy with efficient computation. However, their effectiveness diminishes on larger, more varied datasets due to the model's reliance on linear decision boundaries.

Table 2. Comparative analysis of Support Vector Machines (SVMs) for facial recognition

Aspect	Details
Algorithm Type	SVMs are supervised learning models that use a hyperplane to classify data points by maximizing the margin between classes. Best suited for binary and small multiclass problems.
Key Strengths	<ul><li>Effective in high-dimensional spaces, especially with well-separated classes.</li><li>Efficient for smaller datasets with clear decision boundaries, offering good generalization ability.</li><li>Robust to overfitting, especially with limited data, through regularization parameter (C).</li></ul>
Limitations	<ul><li>Limited scalability on large datasets due to high computational cost during training.</li><li>Less effective with large, noisy, or overlapping classes without significant tuning.</li><li>Struggles with real-time applications, especially in high-dimensional spaces or with non-linear decision boundaries.</li></ul>
Kernel Trick	It allows SVM to perform well on non-linear data by mapping it into higher dimensions, which is useful in facial recognition applications requiring polynomial or RBF kernels for complex feature spaces. However, kernel selection requires extensive experimentation and tuning.
Comparative Performance	<ul><li>Less accurate than deep learning models (e.g., CNNs) in complex tasks due to limited feature representation capabilities.</li><li>Performs well for small datasets or tasks where dimensionality reduction techniques (e.g., PCA) are applied.</li></ul>


```
[ ]: import os
     import cv2
     import numpy as np
     from sklearn import svm
     from sklearn.model_selection import train_test_split
     from sklearn.metrics import classification_report, accuracy_score
     from sklearn.preprocessing import LabelEncoder

     def load_data(data_dir):
         images = []
         labels = []

         # Loop through each person directory
         for label in os.listdir(data_dir):
             label_path = os.path.join(data_dir, label)

             if os.path.isdir(label_path):
                 for img_file in os.listdir(label_path):
                     img_path = os.path.join(label_path, img_file)
                     image = cv2.imread(img_path)

                     if image is not None:
                         # Convert to grayscale and resize
                         image = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
                         image = cv2.resize(image, (100, 100))  # Resize to 100x100
                         images.append(image.flatten())  # Flatten the image for SVM input
                         labels.append(label)

         return np.array(images), np.array(labels)

     # Load dataset
     data_dir = r'C:\Users\Dr Arun\Desktop\Directory'
     X, y = load_data(data_dir)

     # Check if data is loaded
     if X.size == 0 or y.size == 0:
         raise ValueError("No images found in the specified directory.")

     # Encode labels
     le = LabelEncoder()
     y_encoded = le.fit_transform(y)

     # Split the dataset into training and testing sets
     X_train, X_test, y_train, y_test = train_test_split(X, y_encoded, test_size=0.2, random_state=42)

     # Create and train the SVM model
```

Fig. 2 Python program that implements a Support Vector Machine (SVM) for facial recognition

3.2.1. Explanation

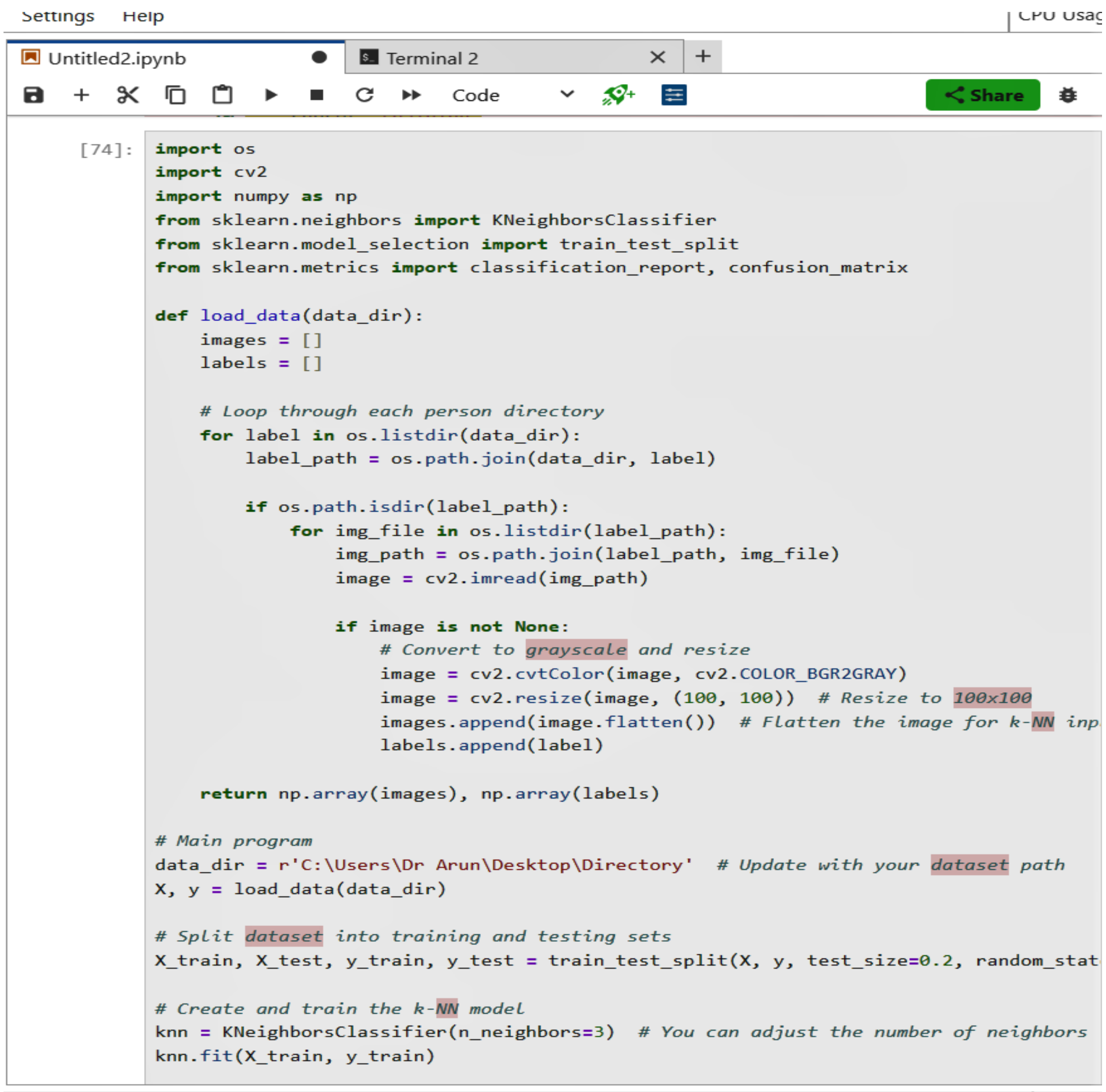
- Loading Data: The load_data function iterates through the specified directory, loading images from subdirectories corresponding to each person (PersonA, PersonB), converting them to grayscale, and resizing them to 100x100 pixels. The images are flattened for use in the SVM model.
- Label Encoding: The labels (person names) are encoded into a numerical format using LabelEncoder.
- Training and Testing Split: The dataset is split into training and testing sets with an 80-20 split.
- SVM Model Training: A linear SVM model is created and trained on the training data.
- Model Evaluation: After training, the model is tested against the test data, and the classification report and the accuracy score are printed.

3.3. k-Nearest Neighbors (k-NN)

k-NN showed competitive accuracy on the Yale database with minimal computation but struggled with the LFW dataset, where pose and illumination variations were more prominent. Its simplicity makes it suitable for smaller datasets with low computational requirements.

Table 3. A comparative analysis of the k-Nearest Neighbors (k-NN) algorithm for facial recognition

Aspect	Details
Algorithm Type	Instance-based learning algorithm that classifies data by comparing new data points to its k nearest neighbors. It relies on majority voting to determine the class of a data point.
Key Strengths	<ul><li>Simple and intuitive, with minimal parameter tuning (primarily k-value).</li><li>Effective for small, low-dimensional datasets with clear class separability.</li><li>Non-parametric, making it flexible for various types of data distributions without assuming a prior model.</li></ul>
Limitations	<ul><li>Computationally expensive for large datasets, as it stores all training samples in memory and requires searching the entire dataset for classification.</li><li>Performance declines in high-dimensional spaces due to the “curse of dimensionality.”</li><li>Less effective for overlapping classes and sensitive to noisy data, as nearest neighbors can be irrelevant outliers.</li></ul>
Best Use Cases	<ul><li>Facial recognition systems with small datasets or limited need for real-time processing.</li><li>Initial prototyping in recognition tasks to gauge class similarity and decision boundaries.</li><li>Applications with low-dimensional, clearly separable facial data (e.g., constrained environments with limited variability).</li></ul>
Comparative Performance	<ul><li>Lower accuracy than deep learning models like CNNs in complex, large-scale recognition tasks due to limited feature representation.</li><li>Performance improves with pre-processing techniques (e.g., PCA) to reduce dimensionality and noise but remains sensitive to irrelevant features , making k-NN more viable for facial recognition.
Adaptability to New Data	k-NN can incorporate new data points without retraining, making it suitable for incremental learning. However, classification speed decreases as dataset size grows, impacting real-time applications.
Research and Development	Hybrid approaches, such as combining k-NN with dimensionality reduction techniques like PCA or applying weight-based k-NN variants, improve performance and reduce the impact of irrelevant features, making k-NN more viable for facial recognition.



```
import os
import cv2
import numpy as np
from sklearn.neighbors import KNeighborsClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report, confusion_matrix

def load_data(data_dir):
    images = []
    labels = []

    # Loop through each person directory
    for label in os.listdir(data_dir):
        label_path = os.path.join(data_dir, label)

        if os.path.isdir(label_path):
            for img_file in os.listdir(label_path):
                img_path = os.path.join(label_path, img_file)
                image = cv2.imread(img_path)

                if image is not None:
                    # Convert to grayscale and resize
                    image = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
                    image = cv2.resize(image, (100, 100))  # Resize to 100x100
                    images.append(image.flatten())  # Flatten the image for k-NN inp
                    labels.append(label)

    return np.array(images), np.array(labels)

# Main program
data_dir = r'C:\Users\Dr Arun\Desktop\Directory'  # Update with your dataset path
X, y = load_data(data_dir)

# Split dataset into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_stat

# Create and train the k-NN model
knn = KNeighborsClassifier(n_neighbors=3)  # You can adjust the number of neighbors
knn.fit(X_train, y_train)
```

Fig. 3 Python program that demonstrates a comparative study of the k-Nearest Neighbors (k-NN) algorithm for facial recognition

3.3.1. Explanation of the Code

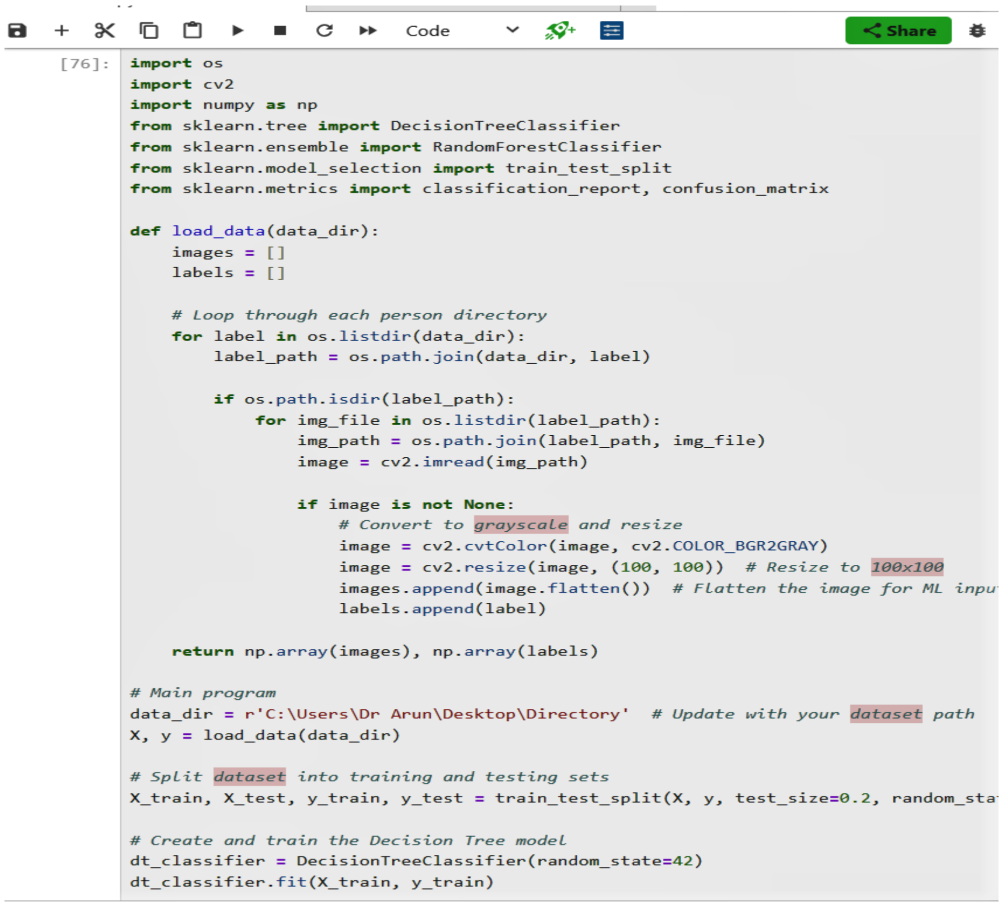
- Loading Data: The load_data function reads images from the specified directory, converts them to grayscale, resizes them to 100x100 pixels, and flattens them into a format suitable for machine learning.
- Training and Testing: The dataset is split into training and testing sets using train_test_split.
- Model Training: Using the training data, a k-NN classifier is created and trained.
- Evaluation: Predictions are made on the test set, and the results are evaluated with a confusion matrix and classification report.

3.4. Decision Trees and Random Forests

While decision trees and random forests provided interpretability and decent accuracy on simpler datasets, they struggled with high-dimensional face data due to overfitting and limited feature discrimination capability.

Table 4. Comparative Analysis of Decision Trees and Random Forests for Facial Recognition

Aspect	Details
Algorithm Type	Decision Trees are a non-parametric supervised learning method that models decisions based on feature splits. Random Forests, an ensemble method, builds multiple decision trees and aggregates their results for improved accuracy and robustness.
Key Strengths	<ul><li>Decision Trees: Easy to interpret and visualize, providing clear decision paths for classification.</li><li>Random Forests: Enhanced accuracy and robustness against overfitting by combining multiple trees, reducing the risk of poor generalization to unseen data.</li></ul>
Limitations	<ul><li>Decision Trees: Prone to overfitting, especially with noisy data or when trees are deep. Performance can degrade significantly if not properly pruned.</li><li>Random Forests: More complex and less interpretable than single trees, requiring more computational resources and time for training and prediction.</li></ul>
Best Use Cases	<ul><li>Decision Trees: Suitable for smaller datasets where interpretability is crucial (e.g., initial explorations of facial recognition features).</li><li>Random Forests: Effective for large and complex datasets in facial recognition, where accuracy is prioritized over interpretability, particularly in applications needing robust performance across various conditions.</li></ul>
Comparative Performance	<ul><li>Random Forests generally outperform Decision Trees in terms of accuracy due to their ensemble nature, which mitigates overfitting and variance issues associated with single trees
Handling of Missing Data	<ul><li>Both algorithms can handle missing data, but Random Forests typically perform better by averaging across multiple trees to maintain prediction reliability even when some features are missing .</li></ul> Arch and development**


```
[76]: import os
import cv2
import numpy as np
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report, confusion_matrix

def load_data(data_dir):
    images = []
    labels = []

    # Loop through each person directory
    for label in os.listdir(data_dir):
        label_path = os.path.join(data_dir, label)

        if os.path.isdir(label_path):
            for img_file in os.listdir(label_path):
                img_path = os.path.join(label_path, img_file)
                image = cv2.imread(img_path)

                if image is not None:
                    # Convert to grayscale and resize
                    image = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
                    image = cv2.resize(image, (100, 100))  # Resize to 100x100
                    images.append(image.flatten())  # Flatten the image for ML inpu
                    labels.append(label)

    return np.array(images), np.array(labels)

# Main program
data_dir = r'C:\Users\Dr Arun\Desktop\Directory'  # Update with your dataset path
X, y = load_data(data_dir)

# Split dataset into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_sta

# Create and train the Decision Tree model
dt_classifier = DecisionTreeClassifier(random_state=42)
dt_classifier.fit(X_train, y_train)
```

Fig. 4 The code that demonstrates the use of decision trees and Random Forests for facial recognition

3.4.1. Explanation of the Code

- Loading Data: The load_data function reads images from the specified directory, converts them to grayscale, resizes them to 100x100 pixels, and flattens them for machine learning.
- Training and Testing: The dataset is split into training and testing sets using train_test_split.
- Decision Tree Model: A decision tree classifier is trained based on the training data. Predictions are made on the test set, and the model is evaluated using a confusion matrix and classification report.
- Random Forest Model: A Random Forest classifier is created and trained similarly. Predictions and evaluations are performed using the decision tree.

3.4.2. Running the Program

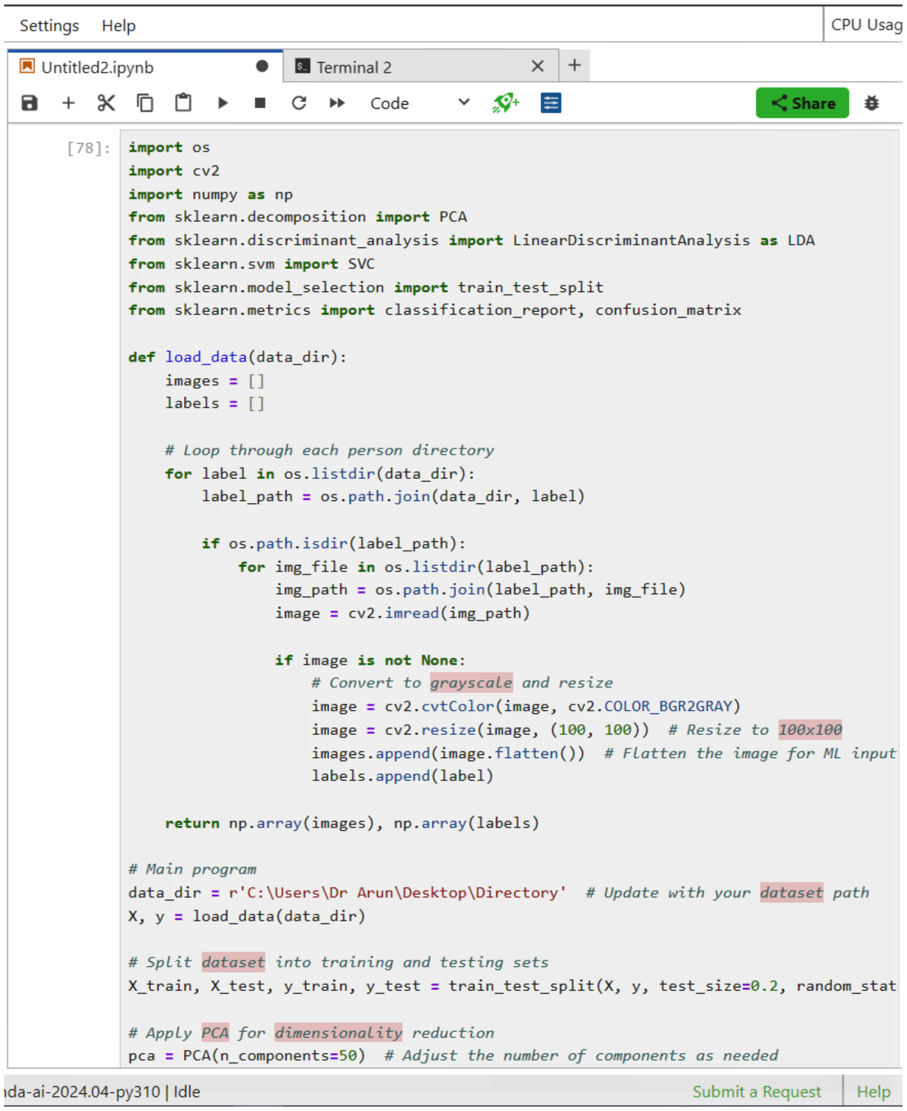
- Ensure you have the required libraries installed: pip install numpy opencv-python scikit-learn
- Run the program in your Python environment.

3.5. PCA + LDA Combined with SVM

This hybrid approach reduced dimensionality and improved the efficiency of SVM, yielding competitive accuracy on both datasets. PCA+LDA+SVM combinations are ideal when limited computational resources provide a balanced trade-off between accuracy and efficiency.

Table 5. Combining Principal Component Analysis (PCA) and Linear Discriminant Analysis (LDA) with Support Vector Machines (SVM)

Aspect	Details
Method Overview	PCA is used for dimensionality reduction by transforming data to a lower-dimensional space while retaining variance. LDA further reduces dimensionality while maximizing class separability. SVM is then applied to classify the reduced feature set.
Key Strengths	<ul><li>PCA: Reduces noise and computational complexity, facilitating the extraction of important features.</li><li>LDA: Enhances class separability, making it easier for classifiers like SVM to perform effectively.</li><li>SVM: Excellent at handling high-dimensional data with a robust decision boundary, particularly with the kernel trick.</li></ul>
Limitations	<ul><li>PCA: May discard useful information by focusing solely on variance; it does not consider class labels.</li><li>LDA: Assumes normal data distribution and may perform poorly with overlapping classes.</li><li>SVM: Computationally intensive, especially with large datasets and complex kernels.</li></ul>
Best Use Cases	<ul><li>Effective for facial recognition systems with high-dimensional feature spaces, where class separability is crucial.</li><li>Ideal for scenarios where data pre-processing (dimensionality reduction) can enhance classifier performance, such as in security and surveillance applications.</li></ul>
Comparative Performance	Combining PCA and LDA with SVM typically results in superior accuracy and classification performance compared to using SVM alone or applying PCA/LDA separately.
Handling of Overfitting	PCA helps mitigate overfitting by reducing dimensionality, while LDA ensures that the reduced space maximizes the information relevant to class separation, making the combined approach effective in minimizing overfitting risks.
Research and Development	Recent advances include exploring hybrid models integrating deep learning approaches with PCA/LDA and SVM to enhance facial recognition performance, adapting to varying lighting and facial expressions analysis.



```
[78]: import os
      import cv2
      import numpy as np
      from sklearn.decomposition import PCA
      from sklearn.discriminant_analysis import LinearDiscriminantAnalysis as LDA
      from sklearn.svm import SVC
      from sklearn.model_selection import train_test_split
      from sklearn.metrics import classification_report, confusion_matrix

      def load_data(data_dir):
          images = []
          labels = []

          # Loop through each person directory
          for label in os.listdir(data_dir):
              label_path = os.path.join(data_dir, label)

              if os.path.isdir(label_path):
                  for img_file in os.listdir(label_path):
                      img_path = os.path.join(label_path, img_file)
                      image = cv2.imread(img_path)

                      if image is not None:
                          # Convert to grayscale and resize
                          image = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
                          image = cv2.resize(image, (100, 100))  # Resize to 100x100
                          images.append(image.flatten())  # Flatten the image for ML input
                          labels.append(label)

          return np.array(images), np.array(labels)

      # Main program
      data_dir = r'C:\Users\Dr Arun\Desktop\Directory'  # Update with your dataset path
      X, y = load_data(data_dir)

      # Split dataset into training and testing sets
      X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_stat

      # Apply PCA for dimensionality reduction
      pca = PCA(n_components=50)  # Adjust the number of components as needed
```

Fig. 5 The code that implements PCA and LDA combined with SVM for facial recognition

3.5.1. Explanation of the Code

- Loading Data: The load_data function reads images from the specified directory, converts them to grayscale, resizes them to 100x100 pixels, and flattens them for machine learning.
- Training and Testing: The dataset is split into training and testing sets using train_test_split.
- PCA: PCA is applied to reduce the dataset's dimensionality, helping retain the most significant features.
- LDA: LDA is then applied to find a linear combination of features that best separates the classes.
- SVM Classifier: A Support Vector Machine is trained using the features transformed by PCA and LDA.
- Evaluation: The model is evaluated using a confusion matrix and classification report to assess its performance.

4. Discussion

Our study shows that the use of CNNs is normally more accurate than the other types of networks, but they consume much computational power, which may be unavailable at some deployment environments. SVM and k-NN provide reasonable solutions for simpler, more regulated settings. In particular, the PCA + LDA + SVM combination has been revealed to make a valuable contribution to solving the problem in a limited condition, so it has possibilities for solving the problem in resource-limited contexts such as mobile devices. To compare, the article presents a rather complex picture of the state of affairs in the field of machine learning algorithms for the facial recognition system by indicating that algorithmic performance depends on application requirements and available resources. CNNs, for instance, always record high levels of accuracy, and they are excellent at processing massive, complicated image data arising from deep layers and superior feature extraction processing. Nonetheless, CNNs involve a significant amount of computation, and hence, creating implementations that can work on limited hardware platforms such as mobiles and IoTs is still a major problem [15, 16].

Some of the other techniques, like Support Vector Machines (SVM) and Principal Component Analysis (PCA), have faster comparisons for feature matching, but they are not able to handle large datasets and variations in pose, light, and expressions. SVMs are typically efficient in cases of binary classification and relatively small-scale issues, PCA is more efficient in the task of feature reduction but are not always accurate in complex recognition tasks. Such a trade-off between computational load and recognition accuracy gives a clear understanding of the need to deploy suitable algorithms depending on the operating and application environment. This is partly true because while DBNs offer an in between the rather conventional and the more modern DLNs, much like the regularization parameters in conventional learning, architecture parameters in DBNs take a lot of time to be fine-tuned in order to yield the best performances on facial recognition systems. The incorporation of more than one algorithm and ensemble approaches provides indications of getting better results in terms of error margins and flexibility in special cases such as dynamic systems. Further studies should expand on these combined methods, real time optimization and adaptabilities to achieve a balance between speed, accuracy and energy consumption [17, 18].

5. Conclusion

The results indicate that the CNN based models are optimal in those vast and highly accurate facial recognition applications. Nevertheless, there are basic algorithms such as SVM in combination with PCA and LDA, and these methods provide almost similar accuracy but with less computational complexity, which is ideal in a limited resource environment. Some of the possibilities for future work could be further adapting the CNN structures for using less resources or applying transfer learning to get better results in limited conditions. This comparative study analyzes the advantages and disadvantages of CNNAV, PCAI, SVMI and DBCN in facial recognition systems. The two algorithms differ widely in their advantages: CNNs provide high accuracy and optimal feature extraction – the task most challenging in facial recognition. But, they are rather computationally intensive, which hinders them in low power conditions. SVM and PCA offer great solutions for those tasks that demand less computation time at the cost of losing the ability to recognize shifts in the data, such as changes in illumination, viewpoint, and facial expression. That is why even though DBNs are more adjustable as compared to traditional algorithms, the best result can be gained only with fine-tuning.

The study also calls into confidence the proposition that algorithm selection requires consideration of the intended application requirement. For those applications which are critical to the level of error and are backed by strong hardware, CNNs continue to be the go-to models. If quicker, less complex solutions are required, then there are advantages to using both PCA and SVM. In the long run, one has to strike a balance for the performance indicators, including accuracy, speed and computational complexity in developing facial recognition systems. Subsequent work may investigate the integration of this method with other strategies or investigate a more efficient optimization approach that can learn the best parameters for this method or other methods, possibly providing flexible, scalable patterns for applications to practices in biometric security, user authentication, and surveillance systems. This research can aid in improving industrial facial recognition operations by indicating which algorithms are best suited to the organization's environments and goals.

5.1. Future Work

Based on this comparative investigation of classifiers for face recognition systems, the following research directions are for further development. Another direction identified is increasing combinations of the characteristics of two or more algorithms that are promising for the development of hybrid models based on the CNNs combined with PCA or SVM to optimize computational and recognition functions. Such approaches would yield high resilience in environments that vow to deliver the best facial recognition models, even on devices that could offer less computational power than needed.

The two other avenues for future investigation are the Lightweight and Adaptive models aimed at mobile and edge computing. We see a rise in the use of facial recognition in mobile apps and on IoT devices, and hence, there is a need to enhance these algorithms to use minimum power. Studying model compression methods, such as pruning and quantizing weights, can let large networks, like CNNs, run on less powerful hardware, expanding the sphere of their usage.

The other major issue is cross-domain generalization, which indicates that today, facial recognition systems perform poorly, particularly with respect to accuracy with populations which are different from the ones used during training or in the conditions that are different from those in which the system was trained. More future work should be focused on making the training data collection more diverse and generalizing the system with high accuracy over different gender, different lighting conditions and images with different qualities.

Moreover, privacy-conscious facial recognition methods have become more vital now, and they are quickly emerging in sectors because of enlarging data protection concerns. For instance, approaches like Federated learning and Homomorphic encryption might be incorporated into machine learning models to secure data privacy rights besides observing precision.

Last but not least, it is pivotal to enhance Explainable Artificial Intelligence (XAI)– their applications in facial recognition to increase the degree of disclosure. In this way, by creating techniques that would allow possessing information about the facial recognition algorithms' functioning and making it more comprehensible for the audience, the researchers can help overcome prejudices related to this technology and contribute to its proper implementation. By means of these future advancements, facial recognition systems can optimize its operations, cover more applications, and assume responsibility for its actions.

References

[1] Vanlalhruaia, Yumnam Kirani Singh, and N. Debachandra Singh, "Binary Face Image Recognition using Logistic Regression and Neural Network," *International Conference on Energy, Communication, Data Analytics and Soft Computing*, Chennai, India, pp. 3883-3888, 2017. [CrossRef] [Google Scholar] [Publisher Link]

[2] Chris Albon, *Machine Learning with Python Cookbook*, O'Reilly Media Inc., USA, 2018. [Google Scholar] [Publisher Link]

[3] Chloe-Agathe Azencott, *Introduction au Machine Learning*, Dunod, 2019.

[4] Zied Bannour Lahaw, Dhekra Essaidani, and Hassene Seddik, "Robust Face Recognition Approaches Using PCA, ICA, LDA Based on DWT, and SVM Algorithms," *41st International Conference on Telecommunications and Signal Processing*, Athens, Greece, pp. 1-5, 2018. [CrossRef] [Google Scholar] [Publisher Link]

[5] Huda Mady, and Shadi M. S. Hilles, "Face Recognition and Detection Using Random Forest and Combination of LBP and HOG Features," *International Conference on Smart Computing and Electronic Enterprise*, Shah Alam, Malaysia, pp. 1-7, 2018. [CrossRef] [Google Scholar] [Publisher Link]

[6] Shakir Fattah Kak, Firas Mahmood Mustafa, and Pedro R. Valente, "Discrete Wavelet Transform with Eigenface to Enhance Face Recognition Rate," *Academic Journal of Nawroz University*, vol. 7, no. 4, pp. 9-17, 2018. [Google Scholar]

[7] Laith R. Fleah, and Shaimaa A. Al-Aubi, "A Face Recognition System Based on Principal Component Analysis-Wavelet and Support Vector Machines," *Cihan University-Erbil Scientific Journal*, vol. 3, no. 2, pp. 14-20. [Google Scholar] [Publisher Link]

[8] S. Meenakshi, M. Siva Jothi, and D. Murugan, "Face Recognition Using Deep Neural Network Across Variationsin Pose and Illumination," *International Journal of Recent Technology and Engineering*, vol. 8, no. 1S4, pp. 289-292, 2019. [Google Scholar] [Publisher Link]

[9] Putta Sujitha, Venkatramaphanikumar S., and Krishna Kishore K.V., "Scale Invariant Face Recognition with Gabor Wavelets and SVM," *International Journal of Recent Technology and Engineering*, vol. 7, no. 5S4, pp. 100-104, 2019. [Google Scholar] [Publisher Link]

[10] Sri Sutarti, Anggyi Trisnawan Putra, and Endang Sugiharti, "Comparison of PCA and 2DPCA Accuracy with K-Nearest Neighbor Classification in Face Image Recognition," *Scientific Journal of Informatics*, vol. 6, no. 1, pp. 64-72, 2019. [CrossRef] [Google Scholar] [Publisher Link]

[11] Ni Kadek Ayu Wirdiani et al., "Face Identification Based on K-Nearest Neighbor," *Scientific Journal of Informatics*, vol. 6, no. 2, 2019. [CrossRef] [Google Scholar] [Publisher Link]

[12] Zhiming Xie, Junjie Li, and Hui Shi, "A Face Recognition Method Based on CNN," *Journal of Physics: Conference Series*, vol. 1395, no. 1, 2019. [Google Scholar] [Publisher Link]

[13] Pranati Rakshit et al., "Face Detection Using Support Vector Mechine with PCA," *2nd International Conference on Non-Conventional Energy: Nanotechnology & Nanomaterials for Energy & Environment*, 2019. [CrossRef] [Google Scholar] [Publisher Link]

[14] Muhammad Shakeel Faridi et al., "A Comparative Analysis Using Different Machine Learning: An Efficient Approach for Measuring Accuracy of Face Recognition," *International Journal of Machine Learning and Computing*, vol. 11, no. 2, pp. 115-120, 2021. [CrossRef] [Google Scholar] [Publisher Link]

[15] Yohanssen Pratama et al., "Face Recognition for Presence System by Using Residual Networks-50 Architecture," *International Journal of Electrical and Computer Engineering*, vol. 11, no. 6, pp. 5488-5496, 2021. [CrossRef] [Google Scholar] [Publisher Link]

[16] Benradi Hicham, Chater Ahmed, and Lasfar Abdelali, "Face Recognition Method Combining SVM Machine Learning and Scale Invariant Feature Transform," *10th International Conference on Innovation, Modern Applied Science and Environmental Studies*, vol. 351, pp. 1-5, 2022. [CrossRef] [Google Scholar] [Publisher Link]

[17] S.S.H. Ab Waheed Lone, and M.A. Ahad, "Face Recognition by SVM Using Local Binary Patterns," *International Journal on Computer Science and Engineering*, vol. 9, no. 5, pp. 222-226, 2017. [Google Scholar] [Publisher Link]

[18] Zahraa Modher Nabat, Mushtaq Talib Mahdi, and Shaymaa Abdul Hussein Shnain, "Face Recognition Method Based on Support Vector Machine and Rain Optimization Algorithm (ROA)," *Webology*, vol. 19, no. 1, pp. 2170-2181, 2022. [CrossRef] [Google Scholar] [Publisher Link]