

Research Article

Exploring Machine Learning and Deep Learning Techniques for Occluded Face Recognition: A Comprehensive Survey and Comparative Analysis

Keny Muhamada ¹, De Rosal Ignatius Moses Setiadi ^{1,2,*}, Usman Sudibyo ^{1,2}, Budi Widjajanto ³, and Arnold Adimabua Ojugo ⁴

- ¹ Informatics Engineering Department, Faculty of Computer Science, Dian Nuswantoro University, Indonesia; e-mail : muhamada.keny@gmail.com; moses@dsn.dinus.ac.id; usman.sudibyo@dsn.dinus.ac.id
 - ² Research Center for Quantum Computing and Materials Informatics, Faculty of Computer Science, Dian Nuswantoro University, Semarang 50131, Indonesia
 - ³ Information System Department, Faculty of Computer Science, Dian Nuswantoro University, Indonesia; e-mail : budi.widjajanto@dsn.dinus.ac.id
 - ⁴ Department of Computer Science, Federal University of Petroleum Resources Effurun, Delta State, Nigeria; e-mail : ojugo.arnold@fupre.edu.ng
- * Corresponding Author : De Rosal Ignatius Moses Setiadi

Abstract: Face recognition occluded by occlusions, such as glasses or shadows, remains a challenge in many security and surveillance applications. This study aims to analyze the performance of various machine learning and deep learning techniques in face recognition scenarios with occlusions. We evaluate KNN (standard and FisherFace), CNN, DenseNet, Inception, and FaceNet methods combined with a pre-trained DeepFace model using three public datasets: YALE, Essex Grimace, and Georgia Tech. The results show that KNN maintains the highest accuracy, reaching 100% on two datasets (Essex Grimace and YALE), even in the presence of occlusions. Meanwhile, CNN shows strong performance, with accuracy remaining 100% on YALE, both with and without occlusions, although its performance drops slightly on Essex Grimace (94% with occlusion). DenseNet and Inception show a more significant drop in accuracy when faced with occlusion, with DenseNet dropping from 81% to 72% on Essex Grimace and Inception dropping from 100% to 92% on the same dataset. FaceNet + DeepFace excels on more large dataset (Georgia Tech) with 98% accuracy, but its performance drops dramatically to 53% and 70% on Essex Grimace and YALE with occlusion. These findings indicate that while deep learning methods show high accuracy under ideal conditions, machine learning methods such as KNN are more flexible and robust to occlusion in face recognition.

Received: August, 29th 2024
Revised: September, 17th 2024
Accepted: September, 25th 2024
Published: September, 26th 2024
Curr. Ver.: September, 26th 2024

Keywords: DeepFace; Face Recognition; FaceNet; FisherFace; Occluded Face Recognition; Obstructed Face Recognition.

1. Introduction

Facial recognition has been proven effective in identifying criminals at the scene with a high level of accuracy, reaching 98% in some cases, making this technology very useful in helping to detect and prevent criminal behavior[1]. Facial recognition not only improves accuracy but also speed in criminal investigations, making it an important tool in modern law enforcement[2]. Facial recognition technology is a form of biometrics based on identifying a person's facial features using facial images and automatic processing devices[3]. This technology is designed to recognize or confirm a person's identity through digital images or videos. Facial recognition has spread to various fields such as public place surveillance, emotion detection, security, social networks, military, attendance, and access control in various industries[4]–[9].

With the rapid development of technology, especially in machine learning and deep learning, various methods have been developed to improve the accuracy and efficiency of facial recognition. These methods include Support Vector Machine (SVM)[10], Convolutional Neural Network (CNN)[10], [11], EigenFace[12], and Principal Component Analysis



Copyright: © 2024 by the authors.
Submitted for possible open access publication under the terms and conditions of the Creative Commons Attribution (CC BY SA) license (<https://creativecommons.org/licenses/by-sa/4.0/>)

(PCA)[13], each of which offers a different approach to identifying and classifying faces based on patterns and features extracted from images [14]. KNN is one of the algorithms often used in face recognition. This algorithm uses the nearest object classification approach to group facial patterns in images by identifying a number of nearest neighbors of an unknown data point and then determining the class of the data based on the majority class of its nearest neighbors[15]. In addition to the KNN method, CNN also plays an important role in computer vision, especially in the application of face recognition, where it can recognize and classify patterns in images with a high degree of accuracy[16], [17].

FisherFace is an algorithm derivative of Fisher Linear Discriminant (FLD) combined with PCA. By combining FLD for class discrimination and PCA for dimensionality reduction, FisherFace can improve efficiency and accuracy in face recognition systems[18]. Inception is a deep learning model in CNN designed to optimize face recognition using one training image per class[19]. DenseNet is a neural network architecture optimized for recognition and classification tasks, where parameter fine-tuning techniques are often used in computer vision to improve performance by pre-training on different datasets before fitting them to the target dataset[20], [21]. FaceNet is a face recognition method that uses deep neural networks and triplet loss training, which has been shown to provide excellent results and superior accuracy compared to other methods[22].

This study explored the performance of the KNN method, FisherFace, and various CNN variants, including DenseNet, Inception, and FaceNet, using the pre-trained DeepFace model for Occluded Face Recognition. The selection of this method was based on a review of previous literature showing the high accuracy of these methods in various face recognition conditions[12], [13], [22]. However, these methods have some limitations. For example, KNN and FisherFace methods, despite showing high accuracy, fail to distinguish between real and occluded facial features. In addition, standard CNNs that do not use pre-trained models have limitations in handling a wider variety of faces, which can lead to overfitting on certain training datasets. The main contributions of this study are:

1. A comprehensive evaluation of various face recognition methods under occluded face conditions has not been widely done in previous studies.
2. A comparative analysis between deep learning methods such as DenseNet and Inception with traditional machine learning methods such as KNN and FisherFace.
3. The application and testing of the FaceNet method with a pre-trained DeepFace model in an occluded face recognition scenario shows varying performance depending on the dataset condition.

The rest of this paper is presented in four sections: Section 2 presents an in-depth literature review of the face recognition methods used. Section 3 describes the experimental procedure and methodology applied in this study. Section 4 discusses the results and analysis, where the performance of each method is analyzed in detail. Section 5 presents conclusions and suggestions for further research, including recommendations for the use of the method in various face recognition applications.

2. Literature Review

2.1 K-Nearest Neighbors (KNN)

K-Nearest Neighbors (KNN) is one of the ten top data mining algorithms using the neighbor-based classification method to reduce classification errors[23], [24]. This algorithm finds the number of K nearest neighbors of an unknown data point based on a certain distance metric, such as Euclidean or Manhattan distance, as shown in Figure 1. KNN is a simple and effective non-parametric technique for pattern recognition in data mining[25]. The performance of KNN is highly dependent on the selection of the right k parameter and distance metric.

KNN is one of the ten most widely used data mining algorithms due to its simplicity and effectiveness in various classification tasks[23]. KNN uses the neighbor-based classification method to determine the class of an unknown data point by finding a number of k nearest neighbors based on a certain distance metric, such as Euclidean distance or Manhattan distance, see Figure 1. KNN is known as a nonparametric technique, which means it does not make explicit assumptions about the data distribution, making it flexible for various types of data and applications [25].

In this study, two variants of KNN were used: standard KNN and FisherFace. Conventional KNN relies on nearest neighbor search based on a specified distance metric, such as Euclidean distance. This approach is practical in many cases but has limitations, especially in face recognition occluded by objects, where the identified features may not be enough to distinguish between real faces and obstructions[26].

FisherFace is a variant of KNN that combines Fisher Linear Discriminant (FLD) with Principal Component Analysis (PCA) to improve classification performance, especially in the case of face recognition. FisherFace uses FLD to maximize the ratio between inter-class and intra-class variability, while PCA is used to reduce the dimensionality of the data before the classification process[27]. By combining these two techniques, FisherFace can capture more relevant discriminative features for face classification, thus providing better accuracy under certain conditions than conventional KNN.

The main difference between standard KNN and FisherFace is how they handle facial features and obstructions. Standard KNN treats all detected features as part of the face without considering whether the feature might be an obstruction. In contrast, FisherFace is more selective in selecting relevant features, as incorporating FLD allows the model to prioritize features with high discriminatory power[28]. In the context of the testing conducted in this study, both KNN variants were evaluated to understand how they handle occluded face recognition. The results of this testing provide insight into the effectiveness of each method under various dataset conditions, which are discussed further in the results and implementation sections.

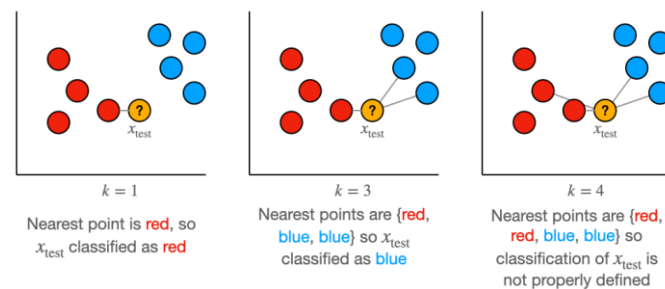


Figure 1. An Illustration of KNN works

2.2 Convolutional Neural Network(CNN)

Convolutional Neural Network (CNN) is one of the most effective deep learning architectures in learning feature extraction for classification purposes and analyzing large amounts of data. CNN is designed to recognize and classify patterns from images through layers of neurons organized in three dimensions, including width, height, and depth. The width and height dimensions describe the spatial aspects of the image, while depth refers to the number of filters or neurons in each layer[29]–[32]. Each layer in CNN is responsible for learning a particular feature, starting from simple features such as edges and textures to more complex features as the network depth increases[33], [34].

In conventional CNN, data flow occurs linearly from one layer to the next. Each convolutional layer flows its output directly to the next layer, allowing for gradual feature learning. However, one of the main challenges of conventional CNN is the vanishing gradient phenomenon, especially in very deep networks. This phenomenon can hinder the training process because the gradient used to update the weights in the network becomes smaller as the number of layers increases[35], [36].

To overcome some of the limitations of conventional CNNs, DenseNet or Densely Connected Convolutional Networks was introduced as a more efficient alternative. DenseNet directly connects each layer to subsequent layers, allowing information and gradients to flow more freely through the network. Instead of relying solely on the previous layer's output, DenseNet combines the outputs of all previous layers as input to each new layer. This approach not only improves the flow of information in the network but also promotes the reuse of previously learned features, which reduces the risk of overfitting and improves network efficiency[37].

Meanwhile, Inception, often referred to as GoogLeNet, introduces the concept of the Inception module, which provides flexibility in choosing the most relevant filter size for

different features in an image. In Inception, each layer combines multiple filter sizes in a single unit, allowing the network to capture different types of features in an image simultaneously. This multi-scale approach allows the model to be more adaptive in dealing with large variations in input data, making it more effective in various applications, including face recognition[38].

Although based on CNN architecture, these three models offer different approaches in the feature extraction process. Conventional CNN follows a linear path in data processing, while DenseNet adopts a more complex connectivity approach to optimize the flow of information. On the other hand, Inception utilizes a variety of filters in each layer to capture more information from the image. This study explores the performance of these three models in the context of Occluded Face Recognition to understand how each model handles the challenge of recognizing faces occluded by objects or certain conditions.

2.3 FaceNet

FaceNet is one of the latest methods in face recognition that uses deep convolutional networks to optimize the representation of faces in the form of feature vectors. This approach is known as one-shot learning (see Figure 2), where the model can be trained using a small number of face images to produce a robust representation. Once the model is trained, it can be used to identify new faces without extensive retraining. FaceNet trains face directly in Euclidean space, where the distance between the resulting feature vectors reflects the similarity between the face models. Smaller distances indicate higher similarity, which allows for easier and more accurate face recognition and classification[22], [39], [40].



Figure 2. Struktur model FaceNet

One of the main advantages of FaceNet is its ability to improve face recognition accuracy by addressing common challenges, such as occlusion, blur, lighting changes, and variations in head pose angles. FaceNet is designed to handle these conditions effectively, enabling accurate face identification even under non-ideal conditions. This method also offers efficiency in terms of the required training data, allowing the model to achieve high performance with a relatively small dataset. In addition, FaceNet's ability to easily update the model without requiring extensive retraining makes it one of the most advanced and efficient face recognition technologies[41].

DeepFace, developed by the Facebook research team, is one of the pre-trained models often combined with FaceNet. DeepFace also uses deep learning to map facial images into 3D space, and gradually trains the network to recognize various facial features through big data. When used in conjunction with FaceNet, DeepFace serves as an initial model that facilitates more effective feature extraction. DeepFace has been tested on various scenarios and has shown high accuracy in various face recognition conditions, making it a popular choice in large-scale face recognition implementations[42], [43]. The combination of FaceNet with the DeepFace model results in a very efficient facial recognition system that can perform well even in difficult conditions such as poor lighting or unusual facial poses. This provides a competitive advantage in face recognition applications, where speed and accuracy are critical.

2.4 Tensorflow

TensorFlow is one of the most widely used open-source libraries for deep learning and machine learning, simplifying and accelerating the research and implementation of neural network models such as CNNs[44]. TensorFlow was developed by a team of researchers at Google Brain as part of an initiative to accelerate progress in machine learning and deep neural network research. The library is designed to make it easier for researchers and practitioners to build, train, and deploy various neural network models, from simple to complex[22].

TensorFlow works based on data flow graphs, where mathematical operations are represented as nodes in the graph, while the graph's edges represent data flows in the form of multidimensional arrays called tensors. Tensors are data structures that allow TensorFlow to handle a wide range of machine learning tasks efficiently, including operations on multidimensional data such as images, text, or signals[45]. Tensors generalize vectors and matrices to higher dimensions, and TensorFlow uses them to store and manipulate data during model training and inference.

One of the main features of TensorFlow is automatic differentiation, which allows automatic gradient computation during the model training process. This is done using back-propagation, where the gradient of the loss function is computed relative to the model parameters to optimize the model performance. The gradient equation optimized through back-propagation can be expressed in Equation (1).

$$\nabla_{\theta} J(\theta) = \frac{\partial J(\theta)}{\partial \theta} \quad (1)$$

Where θ represents the model parameters, and $J(\theta)$ is the loss function. This calculation allows TensorFlow to update the model weights during training efficiently.

TensorFlow also provides a high-level interface like Keras, which makes it easy to build, train, and evaluate neural network models. Keras, originally developed as a separate library, is now an integral part of TensorFlow, allowing users to build models quickly and intuitively without the need to understand deep technical details[46]. With Keras, users can easily define model architecture, loss function, optimizer, and evaluation metrics in a few lines of code. TensorFlow is highly scalable, allowing model execution on various devices ranging from CPUs and GPUs to Tensor Processing Units (TPUs), which Google specifically designs to accelerate deep learning computations[47]. TensorFlow also supports execution on various platforms, including desktops, servers, and mobile devices, making it a top choice for developing machine learning applications that require high performance and flexibility. In addition, TensorFlow has a large ecosystem with additional modules and tools, such as TensorFlow Hub for model reuse, TensorFlow Lite for mobile application development, and TensorFlow.js for implementation in web-based environments. It makes it easy for developers to apply machine learning models in various contexts and applications. This study uses TensorFlow as the main platform to implement and optimize the CNN, DenseNet, and FaceNet models, which have been discussed previously. With its extensive capabilities, TensorFlow enables more efficient and reliable experimentation and development of deep learning models, which are key to achieving accurate results in face recognition.

3. Proposed Method

Several steps must be carried out to carry out face recognition analysis using the specified method, as illustrated in Figure 3. More detailed research stages are explained in sections 3.1 to 3.5.

3.1 Collecting Dataset

The study used three public datasets such as YALE, Essex Grimace, and Georgia Tech. More detailed explanations related to the URL are presented in the data availability statement, while more detailed information about the dataset is presented in Table 1.

3.2 Preprocessing and Data Splitting

It can be seen in Table 1 because the facial image dataset has different types in both Red, Green, and Blue (RGB) color formats, and some use grayscale. Then, all images are converted to grayscale using the OpenCV library (cv2) in Python with the code `resized_image = cv2.resize(image, (180, 200))`. In addition, the image dimensions are also different, so the image needs to be changed to 180×200 pixels. This step aims to align all images in each dataset so that they are the same size and more focused on identifying faces in the image. Furthermore, the dataset is split into three parts, namely Training Data of 80%, Validation Data of 10%, and Testing Data of 10% of the total dataset.

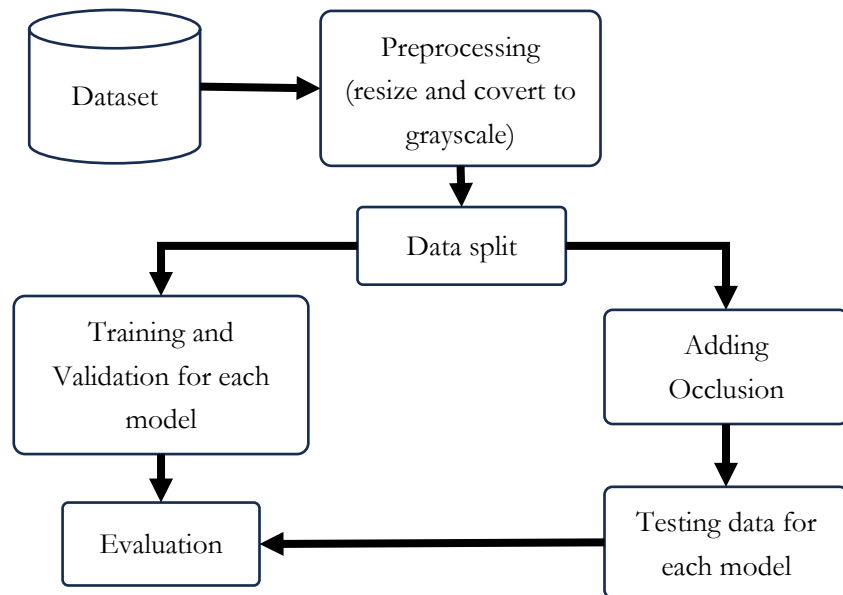
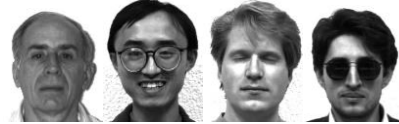




Figure 3. Occluded Face Recognition Method

Table 1. Dataset properties and sample images.

Dataset	Number of Class	Number of Images	Image Dimension	Format	Sample Images
YALE	15	164	320×243	Grayscale	
Essex Grimace	18	360	180×200	RGB	
Georgia Tech	50	750	180×200	RGB	

3.3 Recognition Modelling

Next, the training data is trained using several methods, such as standard KNN and KNN with FisherFace, CNN, DenseNet, Inception, and FaceNet with DeepFace. After going through the data preprocessing section, the next stage is to design a model using the specified methods and architecture..

3.3.1 K-Nearest Neighbors (KNN) Model

In the K-Nearest Neighbors (KNN) method, classification is performed by calculating the distance between an unknown data point and all data points in the training dataset using a specific distance metric. Euclidean Distance is the most commonly used distance metric, calculated using Equation (2).

$$d(p, q) = \sqrt{\sum_{i=1}^n (p_i - q_i)^2} \quad (2)$$

Where p and q are two points in the feature space; p_i and q_i are the i^{th} components of points p and q , respectively; n is the number of features used in the data representation.

Additionally, the FisherFace method, which involves a cascading approach of Principal Component Analysis (PCA) and Linear Discriminant Analysis (LDA), is applied to enhance the performance of the KNN model for face recognition tasks. In the first stage of FisherFace, PCA is utilized to reduce the dimensionality of the facial dataset by identifying and retaining the most significant components that capture the majority of the variance in the data. This step effectively handles features that may be highly correlated, thus simplifying the

data and mitigating the risk of overfitting. By reducing the number of dimensions, PCA helps focus the subsequent analysis on the most relevant features, making the model more efficient.

Following PCA, LDA is applied to the reduced-dimensional data as part of the FisherFace method. The role of LDA in this cascading process is to project the data onto a new axis that maximizes the separation between classes, considering the class labels. LDA optimizes class discrimination by finding the linear combinations of the input features that best separate the different classes. This combination of PCA and LDA within the FisherFace framework ensures that the data is not only simplified but also structured in a way that enhances the performance of the KNN classifier in recognizing faces. Both PCA and LDA are implemented using the Scikit-learn (SkLearn) library, a popular Python library for machine learning, which provides robust tools for executing these dimensionality reduction and classification tasks.

3.3.2 Convolutional Neural Network (CNN) Model

The Convolutional Neural Network (CNN) model consists of several consecutive layers, namely convolutional, pooling, and fully connected layers. The design of this model aims to improve classification accuracy by utilizing important features extracted from the input images [29], [48], [49]. In the first layer, an image of a dimension of 180x200 pixels is input into the network. The first layer is a convolutional layer with 32 filters of size 3x3, as expressed in Equation (3).

$$Z = W * X + b \quad (3)$$

Where W is the filter weight (weight matrix applied to the input image); X is the input (data representation of the image); b is the bias (scalar value added to adjust the output of the convolutional operation).

The output of this convolutional operation is then processed through the Rectified Linear Unit (ReLU) activation function, which is expressed in Equation (4)

$$ReLU(x) = \max(0, x) \quad (4)$$

Next, the output from the first convolutional layer is passed to a pooling (MaxPooling) layer with a size of 2x2, which aims to reduce the spatial dimensions of the feature map. This process is repeated twice with increasing filters: 64 in the second convolutional layer and 128 in the third. A MaxPooling layer of the same size follows each convolutional layer. This process aims to extract more complex and relevant features from the input image.

After the three convolutional and pooling layers, the output is flattened into a one-dimensional (1D) vector to be fed into the dense layer. The output layer is a dense layer with 50 neurons, corresponding to the number of classes to be predicted, using the softmax activation function to produce classification probabilities. The softmax function is expressed in Equation (5). This function transforms the output into a probability distribution for each class, ensuring that the total probability for all classes equals 1.

$$Softmax(x_i) = \frac{e^{x_i}}{\sum_j e^{x_j}} \quad (5)$$

Where x_i is the output of the i^{th} neuron; e^{x_i} is the exponential of the value x_i ; $\sum_j e^{x_j}$ is the sum of all exponential values for all neurons in the output layer.

In addition to CNN, DenseNet is also employed as one of the fine-tuned models using the facial dataset. The pre-trained weights on the ImageNet dataset are leveraged to accelerate convergence and enhance model performance. The output layer is replaced, and an additional dense layer is added for facial identity classification. During training, callbacks such as EarlyStopping and ModelCheckpoint are implemented to optimize the training process and prevent overfitting. Similarly, the Inception model utilizes pre-trained weights from ImageNet to expedite convergence and improve performance. Like DenseNet, Inception also benefits from using callbacks such as EarlyStopping and ModelCheckpoint to ensure the model is effectively optimized and overfitting is minimized.

3.3.3 FaceNet

FaceNet utilizes deep convolutional networks and triplet loss training to maximize the Euclidean distance between the embedding vectors of different facial images, while

minimizing the distance between the [39], [40]embedding vectors of images of the same face. The Triplet Loss formula can be defined in Equation (6).

$$L = \sum_i^N [\|f(x_i^a) - f(x_i^p)\|_2^2 - \|f(x_i^a) - f(x_i^n)\|_2^2 + \alpha] \quad (6)$$

Where x_i^a is the anchor, which is the reference image; x_i^p is the positive, which is the image of the same person as the anchor; x_i^n is the negative, which is the image of a different person from the anchor; $f(x)$ is the embedding of image x ; α is a predefined margin used to prevent the model from making the distance between the anchor and positive too small or the distance between the anchor and negative too large.

FaceNet is configured using the DeepFace library. The FaceNet model is loaded with the command `DeepFace.build_model("Facenet")` internally configures the model for feature extraction processes. DeepFace also provides various functions for further adjustments, including model parameter settings, which the library automatically manages. Overall, the main parameter used in the configuration of FaceNet in this study is the specification of the FaceNet model itself through the DeepFace command, with further adjustments handled by DeepFace to ensure that the model functions optimally in the context of facial feature extraction.

3.4 Adding Occlusion

After the data was divided into three sets (training, validation, and testing), some images within the Testing Data sets added a black object to the eye area. The addition of this black object was designed to mimic the effect of eyeglasses, allowing for an evaluation of how such occlusions affect the model's accuracy in facial recognition[50]. By comparing the accuracy between images with and without the black object over the eyes, the model's robustness to common visual variations can be better understood, as illustrated in Figure 3.



Figure 3. Sample of occlusion effect adding on eyes

3.5 Evaluation

In this section, the performance of the face recognition models is evaluated using accuracy, precision, recall, and F1-score. Accuracy is emphasized as the primary metric due to the nearly balanced class distribution in the dataset, making it a reliable measure of overall model performance[51]. A balanced dataset ensures accuracy reflects how well the models distinguish between different classes without being skewed by class imbalances. While accuracy offers a straightforward assessment of classification effectiveness, precision, recall, and F1-score provide additional insights[48], [52], [53]:

- Precision measures the model's ability to correctly identify positive instances, focusing on minimizing false positives.
- Recall evaluates the model's effectiveness in detecting all relevant positive instances, thereby reducing false negatives.
- F1-score combines precision and recall, offering a balanced metric that accounts for both types of errors.

Although precision, recall, and F1-score contribute to a nuanced understanding of model performance, the focus remains on accuracy, as it offers a clear and comprehensive indication of the models' efficacy, particularly given the balanced nature of the dataset.

4. Results and Discussion

In this section, the performance of the proposed methods is evaluated on three different datasets using the techniques described in detail in the previous section. The results obtained from this evaluation provide insight into the effectiveness of each method under various conditions. The configurations of the CNN and FaceNet models, which play a crucial role in the accuracy and robustness of the face recognition task, are presented in the following table. These configurations highlight the specific parameters and settings used to optimize the performance of the models on each dataset. In the KNN model, the value of $k = 3$ is used, the selection of this value is based on the best results of trial and error, while the configurations of the deep learning model are presented in Table 2.

Table 2. CNN-based and FaceNet Configuration.

Configuration	Parameter Value			
	CNN standard	DenseNet	Inception	FaceNet
Optimizer	adam	adam	adam	-
Loss Function	categorical_crossentropy	categorical_crossentropy	categorical_crossentropy	Triplet Loss
Metrics	Accuracy	Accuracy	Accuracy	-
Callback	-	EarlyStopping (monitor='val_loss', patience=3, verbose=1) ModelCheckpoint(save_best_only)	EarlyStopping (monitor='val_loss', patience=3, verbose=1) ModelCheckpoint(save_best_only)	-
Epoch(s)	20	20	20	-
Feature Extraction	-	-	-	DeepFace.represent
Classifier	softmax	softmax	softmax	SVC(kernel='linear')

Table 3. Accuracy of face recognition result for all methods with and without occlusion

Method	Occlusion	Essex Grimace	YALE	Georgia Tech
KNN	No	100	97	74
	Yes	100	97	-
KNN+ FisherFace	No	100	97	82
	Yes	100	93	-
CNN	No	100	100	82
	Yes	94	100	-
DenseNet	No	81	93	65
	Yes	72	90	-
Inception	No	100	83	91
	Yes	92	53	-
FaceNet + DeepFace	No	97	100	98
	Yes	53	70	-

Based on Table 3, it can be seen that the performance of the face recognition method before and after the addition of occlusion shows a significant pattern. KNN, both in the standard version and when combined with FisherFace, maintains high accuracy. For example, KNN achieves 100% accuracy on the Essex Grimace and YALE datasets, even when there is occlusion. This shows that the KNN method, although a traditional machine learning algorithm, has excellent flexibility in detecting faces with or without occlusion. This advantage could be because KNN relies on the instance-based learning method, which is less affected by changes caused by occlusion.

CNNs also show strong performance, especially in the absence of occlusion. On the Essex Grimace dataset, CNN achieves 100% accuracy without occlusion, but drops slightly

to 94% when occlusion is present. On the YALE dataset, CNN consistently maintains 100% accuracy with and without occlusion, demonstrating the model’s robustness to data variation. CNN, which has a simpler architecture than DenseNet and Inception, appears to be more flexible in handling occlusion, possibly due to its lack of complexity, which allows the model to better adapt to changes.

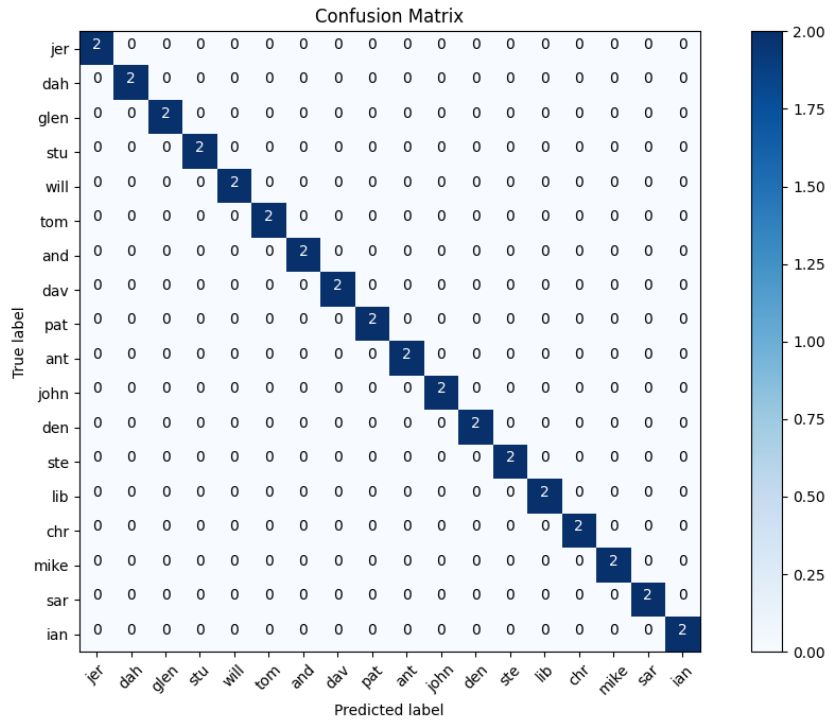


Figure 4. Essex Grimface's best result with occlusion using KNN

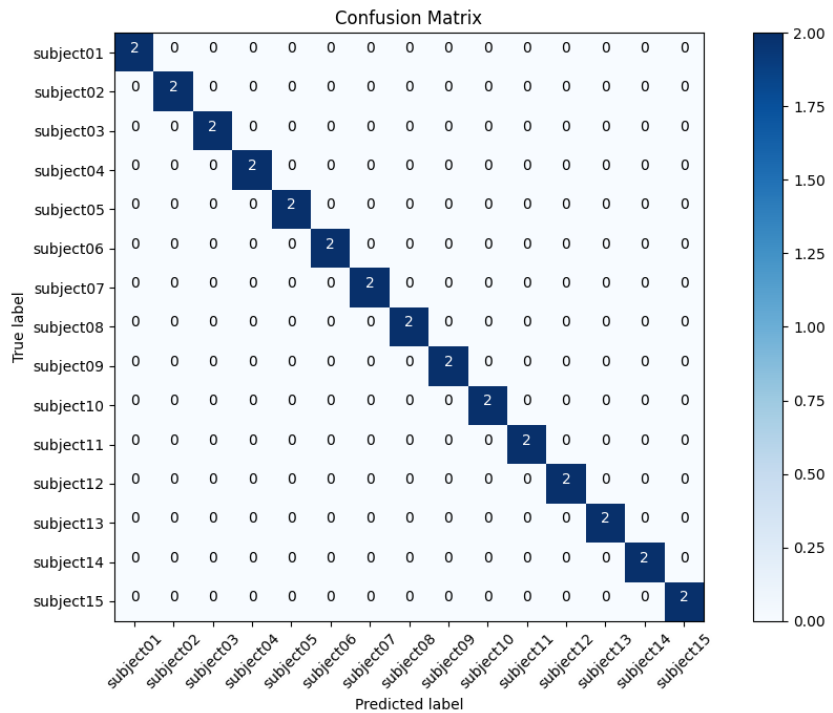


Figure 5. Yale's best result with occlusion using CNN

Meanwhile, DenseNet and Inception show a more significant drop in performance when occlusion is added. DenseNet, for example, drops from 81% to 72% on the Essex Grimace

dataset. Inception also sees a drop in performance from 100% to 92% on the same dataset. This may indicate that these models, which are more complex and typically require more structured data, are more sensitive to perturbations such as occlusion.

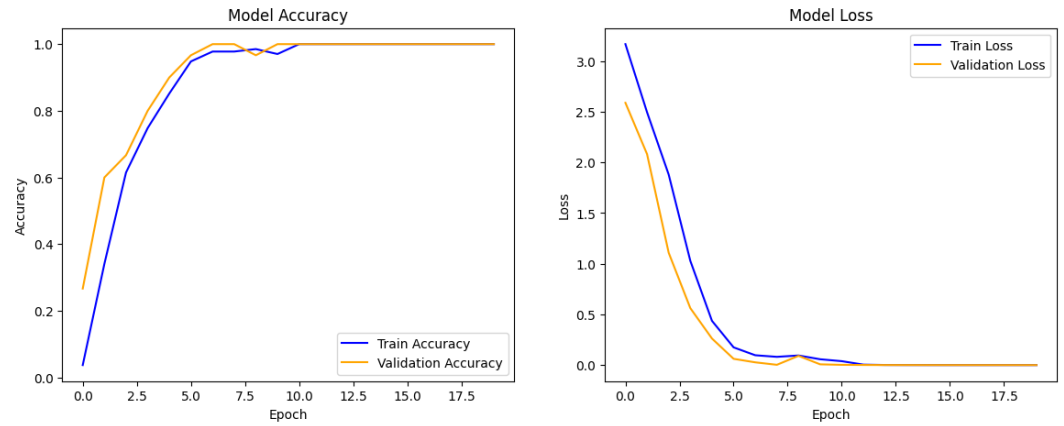


Figure 6. Training and validation accuracy and loss plot on Yale with occlusion using CNN

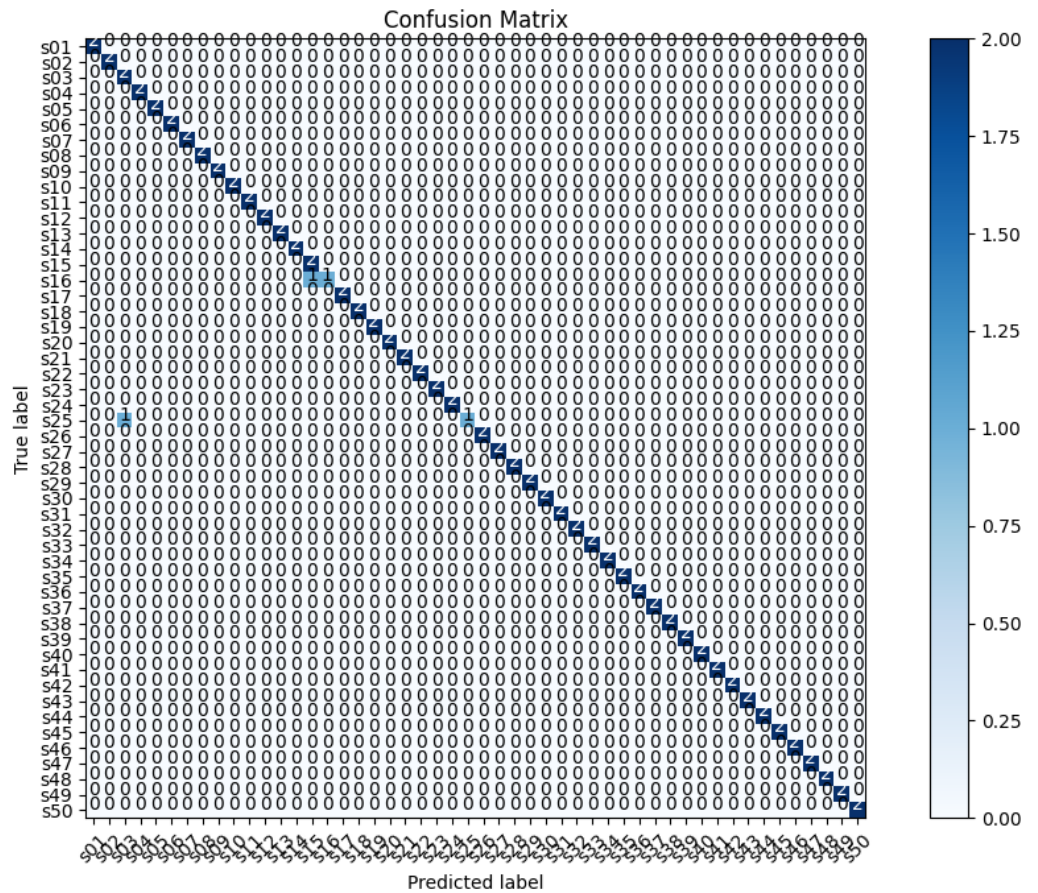


Figure 7. Georgia Tech's best result without occlusion using FaceNet+DeepFace

FaceNet + DeepFace performs the highest on the large Georgia Tech dataset with 98% accuracy, demonstrating its strength in handling large and complex data. However, its performance drops significantly when occlusion occurs, from 97% to 53% on the Essex Grimace dataset and from 100% to 70% on the YALE dataset. While FaceNet + DeepFace is very effective on large, unperturbed datasets, it is less robust in real-world situations where occlusion is typical. Overall, this analysis suggests that traditional machine learning methods such as KNN can offer better robustness to occlusion than more complex deep learning methods.

On the other hand, deep learning models such as CNN, DenseNet, and Inception show a more outstanding performance drop when faced with occlusion, with FaceNet + DeepFace showing the highest sensitivity to perturbation despite outperforming on large datasets. Furthermore, the best confusion matrix results for each dataset are presented in Figures 4, 5, and 7. In addition, the CNN epoch plot is also presented in Figure 6. Based on the confusion matrix display, it can also be seen that the best precision, recall, and F1 values for the Essex Grimface and Yale datasets are 1.0, while for the Georgia Tech data, the precision is 0.99, while the recall and f1 are 0.98. Furthermore, several comparisons were also carried out with several related studies, the results of which are presented in Table 4.

Table 4. Accuracy (%) comparison with related works without occlusion

Method	Essex Grimface	Yale	Georgia Tech
Ref[10]	-	72	-
Ref[17]	-	-	94
Ref[54]	95	-	-
Ours (best results)	100	100	98

Based on the results presented in Table 4, the proposed method appears to perform better than the method for related work without occlusion. This shows that the method analyzed in this study has robust performance and performs image recognition.

5. Conclusions

This study provides an in-depth analysis of the effectiveness of various machine learning and deep learning techniques for face recognition under occlusion. The results show that traditional machine learning methods, such as K-Nearest Neighbors (KNN), demonstrate remarkable robustness in dealing with occlusion, with high accuracy, even in challenging scenarios. In contrast, deep learning models such as CNN, DenseNet, Inception, and FaceNet, while generally achieving high accuracy under ideal conditions, show significant performance degradation when faced with visual occlusion.

Specifically, KNN, in its standard form and when combined with FisherFace, consistently achieves 100% accuracy on the Essex Grimace and YALE datasets, regardless of the presence or absence of occlusion. This suggests that more straightforward instance-based learning methods may offer greater flexibility in real-world applications with common data imperfections. On the other hand, deep learning models, despite their sophisticated architectures, are more sensitive to variations such as occlusion, which can significantly impact their performance. For example, FaceNet + DeepFace, which achieved a peak accuracy of 98% on the large Georgia Tech dataset, saw a significant drop in accuracy when occlusion was introduced, with the score dropping to 53% on the Essex Grimace dataset.

These findings suggest that while deep learning methods excel in environments where data is clean and well-structured, traditional machine learning techniques such as KNN may provide more consistent and reliable performance in practical scenarios where occlusion or noise is present. Future research could explore hybrid approaches that combine the strengths of both methods to develop more robust facial recognition systems that can adapt to various real-world conditions.

Author Contributions: Conceptualization: K.M. and D.R.I.M.S.; Methodology: K.M. and D.R.I.M.S.; Software: K.M.; Validation: U.S., B.W. and A.A.O.; Formal analysis: K.M.; Investigation: All; Resources: K.M. and D.R.I.M.S.; Data curation: K.M.; Writing—original draft preparation: K.M.; Writing—review and editing: All; Visualization: K.M. and D.R.I.M.S.; Supervision: D.R.I.M.S.; Project administration: K.M.; Funding acquisition: All.

Funding: This research received no external funding.

Data Availability Statement: YALE face <https://www.kaggle.com/datasets/olgabelitetskaya/yale-face-database>; Essex Grimace :

<http://cswww.essex.ac.uk/mv/allfaces/grimace.zip>; Georgia Tech :
<https://academictorrents.com/details/0848b2c9b40e49041eff85ac4a2da71ae13a3e4f>

Conflicts of Interest: The authors declare no conflict of interest.

References

- [1] A. Verma, "Crime Detection System Using Face Recognition," *Indian J. Appl. Res.*, pp. 40–42, Sep. 2022, doi: 10.36106/ijar/8212604.
- [2] K. P. Teja, G. D. Kumar, and T. P. Jacob, "Face Detection and Recognition for Criminal Identification," in *2023 8th International Conference on Communication and Electronics Systems (ICCES)*, Jun. 2023, pp. 1431–1435. doi: 10.1109/ICCES57224.2023.10192845.
- [3] L. Li, X. Mu, S. Li, and H. Peng, "A Review of Face Recognition Technology," *IEEE Access*, vol. 8, pp. 139110–139120, 2020, doi: 10.1109/ACCESS.2020.3011028.
- [4] R. Sithara and R. Rajasree, "A survey on Face Recognition Technique," in *2019 IEEE International Conference on Innovations in Communication, Computing and Instrumentation (ICCI)*, Mar. 2019, pp. 189–192. doi: 10.1109/ICCI46240.2019.9404387.
- [5] D. Balakrishnan, U. Mariappan, S. Dharani, Y. Rajyalakshmi, and S. Sravani, "A face recognition and intelligent home automation system," in *International Conference on Mathematics and its Applications in Technology*, 2024, p. 030015. doi: 10.1063/5.0224836.
- [6] K. Madhushree et al., "Development of a highly capable robotic dog with enhanced sensory and communication abilities for military applications," in *International Conference on Condensed Matter and Applied Physics*, 2024, p. 140029. doi: 10.1063/5.0225346.
- [7] M. Eliazar, Y. Mudgil, and D. G. Bachu, "Smart CCTV camera surveillance system," in *4th International Conference on Internet of Things*, 2024, p. 020132. doi: 10.1063/5.0217281.
- [8] A. Y. Felix, S. Vignesh, and M. J. Andrew, "Mobile application and QR code for attendance system," in *4th International Conference on Internet of Things*, 2024, p. 020236. doi: 10.1063/5.0217048.
- [9] A. Pathirana, D. K. Rajakaruna, D. Kasthurirathna, A. Atukorale, R. Aththidiye, and M. Yatiipansalawa, "A Reinforcement Learning-Based Approach for Promoting Mental Health Using Multimodal Emotion Recognition," *J. Futur. Artif. Intell. Technol.*, vol. 1, no. 2, pp. 124–141, 2024, doi: 10.62411/faith.2024-22.
- [10] R. Ravi, S. . Yadhukrishna, and R. Prithviraj, "A Face Expression Recognition Using CNN & LBP," in *2020 Fourth International Conference on Computing Methodologies and Communication (ICCMC)*, Mar. 2020, pp. 684–689. doi: 10.1109/ICCMC48092.2020.ICCMC-000127.
- [11] R. Gupta, A. S. Shekhawat, M. S. Anand, and P. Selvaraj, "Face expression recognition master using convolutional neural network," in *4th International Conference on Internet of Things*, 2024, p. 020069. doi: 10.1063/5.0217855.
- [12] I. U. W. Mulyono, D. R. I. M. Setiadi, A. Susanto, E. H. Rachmawanto, A. Fahmi, and Muljono, "Performance Analysis of Face Recognition using Eigenface Approach," in *Proceedings - 2019 International Seminar on Application for Technology of Information and Communication: Industry 4.0: Retrospect, Prospect, and Challenges, iSemantic 2019*, Sep. 2019, pp. 12–16. doi: 10.1109/ISEMANTIC.2019.8884225.
- [13] G. E. Rani, S. M. M. P. Suresh, M. Abhiram, K. J. Surya, and B. Y. A. N. Kumar, "Face Recognition Using Principal Component Analysis," in *2022 2nd International Conference on Advance Computing and Innovative Technologies in Engineering (ICACITE)*, Apr. 2022, pp. 932–936. doi: 10.1109/ICACITE53722.2022.9823434.
- [14] Y. Kortli, M. Jridi, A. Al Falou, and M. Atri, "Face Recognition Systems: A Survey," *Sensors*, vol. 20, no. 2, p. 342, Jan. 2020, doi: 10.3390/s20020342.
- [15] D. Kurniadi, A. Sugiyono, and L. A. Wardaya, "Pattern Recognition of Human Face With Photos Using KNN Algorithm," *J. Transform.*, vol. 19, no. 1, p. 17, Jul. 2021, doi: 10.26623/transformatika.v19i1.3581.
- [16] Z. Li, F. Liu, W. Yang, S. Peng, and J. Zhou, "A Survey of Convolutional Neural Networks: Analysis, Applications, and Prospects," *IEEE Trans. Neural Networks Learn. Syst.*, vol. 33, no. 12, pp. 6999–7019, Dec. 2022, doi: 10.1109/TNNLS.2021.3084827.
- [17] K. Gowri and B. L. Shivakumar, "Dwindling DCNN Training Time and Improving Authentication by Spurring Algorithm in Cloud Computing," *Int. J. Intell. Eng. Syst.*, vol. 17, no. 3, pp. 503–514, Jun. 2024, doi: 10.22266/ijies2024.0630.39.
- [18] P. Ghadekar, M. Ranjan Pradhan, D. Swain, and B. Acharya, "EmoSecure: Enhancing Smart Home Security With FisherFace Emotion Recognition and Biometric Access Control," *IEEE Access*, vol. 12, pp. 93133–93144, 2024, doi: 10.1109/ACCESS.2024.3423783.
- [19] L. Patil and V. D. Mytri, "Face Recognition with CNN and Inception Deep Learning Models," *Int. J. Recent Technol. Eng.*, vol. 8, no. 3, pp. 1932–1938, Sep. 2019, doi: 10.35940/ijrte.C4476.098319.
- [20] A. Waheed, M. Goyal, D. Gupta, A. Khanna, A. E. Hassaniien, and H. M. Pandey, "An optimized dense convolutional neural network model for disease recognition and classification in corn leaf," *Comput. Electron. Agric.*, vol. 175, p. 105456, Aug. 2020, doi: 10.1016/j.compag.2020.105456.
- [21] N. Becherer, J. Pecarina, S. Nykl, and K. Hopkinson, "Improving optimization of convolutional neural networks through parameter fine-tuning," *Neural Comput. Appl.*, vol. 31, no. 8, pp. 3469–3479, Aug. 2019, doi: 10.1007/s00521-017-3285-0.
- [22] I. William, D. R. Ignatius Moses Setiadi, E. H. Rachmawanto, H. A. Santoso, and C. A. Sari, "Face Recognition using FaceNet (Survey, Performance Test, and Comparison)," in *Proceedings of 2019 4th International Conference on Informatics and Computing, ICIC 2019*, Oct. 2019. doi: 10.1109/ICIC47613.2019.8985786.
- [23] S. Zhang, "Cost-sensitive KNN classification," *Neurocomputing*, vol. 391, pp. 234–242, May 2020, doi: 10.1016/j.neucom.2018.11.101.
- [24] M. A. Araaf, K. Nugroho, and D. R. I. M. Setiadi, "Comprehensive Analysis and Classification of Skin Diseases based on Image Texture Features using K-Nearest Neighbors Algorithm," *J. Comput. Theor. Appl.*, vol. 1, no. 1, pp. 31–40, Sep. 2023, doi: 10.33633/jcta.v1i1.9185.

- [25] J. Gou, W. Qiu, Z. Yi, Y. Xu, Q. Mao, and Y. Zhan, "A Local Mean Representation-based K -Nearest Neighbor Classifier," *ACM Trans. Intell. Syst. Technol.*, vol. 10, no. 3, pp. 1–25, May 2019, doi: 10.1145/3319532.
- [26] R. O. Duda, P. E. Hart, and D. G. Stork, *Pattern Classification*, 2nd ed. Nashville, TN: John Wiley & Sons, 2000.
- [27] P. N. Belhumeur, J. P. Hespanha, and D. J. Kriegman, "Eigenfaces vs. Fisherfaces: recognition using class specific linear projection," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 19, no. 7, pp. 711–720, Jul. 1997, doi: 10.1109/34.598228.
- [28] M. Turk and A. Pentland, "Eigenfaces for Recognition," *J. Cogn. Neurosci.*, vol. 3, no. 1, pp. 71–86, Jan. 1991, doi: 10.1162/jocn.1991.3.1.71.
- [29] M. S. Sunarjo, H. Gan, and D. R. I. M. Setiadi, "High-Performance Convolutional Neural Network Model to Identify COVID-19 in Medical Images," *J. Comput. Theor. Appl.*, vol. 1, no. 1, pp. 19–30, Aug. 2023, doi: 10.33633/jcta.v1i1.8936.
- [30] H. T. Adityawan, O. Farroq, S. Santosa, H. M. M. Islam, M. K. Sarker, and D. R. I. M. Setiadi, "Butterflies Recognition using Enhanced Transfer Learning and Data Augmentation," *J. Comput. Theor. Appl.*, vol. 1, no. 2, pp. 115–128, Nov. 2023, doi: 10.33633/jcta.v1i2.9443.
- [31] M. T. H. Khan Tusar, M. T. Islam, A. H. Sakil, M. N. H. N. Khandaker, and M. M. Hossain, "An Intelligent Telediagnosis of Acute Lymphoblastic Leukemia using Histopathological Deep Learning," *J. Comput. Theor. Appl.*, vol. 2, no. 1, pp. 1–12, May 2024, doi: 10.62411/jcta.10358.
- [32] S. Fanijo, "AI4CRC: A Deep Learning Approach Towards Preventing Colorectal Cancer," *J. Futur. Artif. Intell. Technol.*, vol. 1, no. 2, pp. 143–159, 2024, doi: 10.62411/faith.2024-28.
- [33] A. Amorim, T. Morehouse, D. Kasilingam, R. Zhou, and N. Magotra, "CNN-based AMC for Internet of Underwater Things," in *2021 IEEE International Midwest Symposium on Circuits and Systems (MWSCAS)*, Aug. 2021, pp. 688–691. doi: 10.1109/MWSCAS47672.2021.9531853.
- [34] T. Huynh-The, C.-H. Hua, Q.-V. Pham, and D.-S. Kim, "MCNet: An Efficient CNN Architecture for Robust Automatic Modulation Classification," *IEEE Commun. Lett.*, vol. 24, no. 4, pp. 811–815, Apr. 2020, doi: 10.1109/LCOMM.2020.2968030.
- [35] S. Ioffe and C. Szegedy, "Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift," *arXiv*. Feb. 10, 2015. [Online]. Available: <http://arxiv.org/abs/1502.03167>
- [36] F. M. Firnando, D. R. I. M. Setiadi, A. R. Muslikh, and S. W. Iriananda, "Analyzing InceptionV3 and InceptionResNetV2 with Data Augmentation for Rice Leaf Disease Classification," *J. Futur. Artif. Intell. Technol.*, vol. 1, no. 1, pp. 1–11, May 2024, doi: 10.62411/faith.2024-4.
- [37] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, "Densely Connected Convolutional Networks," in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jul. 2017, pp. 2261–2269. doi: 10.1109/CVPR.2017.243.
- [38] C. Szegedy *et al.*, "Rethinking the Inception Architecture for Computer Vision," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun. 2016, pp. 2818–2826. doi: 10.1109/CVPR.2016.308.
- [39] M. S. Uddin and W. Slany, "Visual Programming for Human Detection Using FaceNet in Pocket Code," *Int. J. Interact. Mob. Technol.*, vol. 18, no. 13, pp. 179–194, Jul. 2024, doi: 10.3991/ijim.v18i13.49277.
- [40] J. A. Mensah, J. K. Appati, E. K. Boateng, E. Ocran, and L. Asiedu, "FaceNet recognition algorithm subject to multiple constraints: Assessment of the performance," *Sci. African*, vol. 23, p. e02007, Mar. 2024, doi: 10.1016/j.sciaf.2023.e02007.
- [41] H.-C. Li, Z.-Y. Deng, and H.-H. Chiang, "Lightweight and Resource-Constrained Learning Network for Face Recognition with Performance Optimization," *Sensors*, vol. 20, no. 21, p. 6114, Oct. 2020, doi: 10.3390/s20216114.
- [42] Y. Taigman, M. Yang, M. Ranzato, and L. Wolf, "DeepFace: Closing the Gap to Human-Level Performance in Face Verification," in *2014 IEEE Conference on Computer Vision and Pattern Recognition*, Jun. 2014, pp. 1701–1708. doi: 10.1109/CVPR.2014.220.
- [43] H. Phan, C. X. Le, V. Le, Y. He, and A. T. Nguyen, "Fast and Interpretable Face Identification for Out-Of-Distribution Data Using Vision Transformers," in *2024 IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, Jan. 2024, pp. 6289–6299. doi: 10.1109/WACV57701.2024.00618.
- [44] B. Pang, E. Nijkamp, and Y. N. Wu, "Deep Learning With TensorFlow: A Review," *J. Educ. Behav. Stat.*, vol. 45, no. 2, pp. 227–248, Apr. 2020, doi: 10.3102/1076998619872761.
- [45] T. X. Truong *et al.*, "A New Approach Based on TensorFlow Deep Neural Networks with ADAM Optimizer and GIS for Spatial Prediction of Forest Fire Danger in Tropical Areas," *Remote Sens.*, vol. 15, no. 14, p. 3458, Jul. 2023, doi: 10.3390/rs15143458.
- [46] F. Chollet, *Deep learning with python*. New York, NY: Manning Publications, 2022.
- [47] N. P. Jouppi *et al.*, "In-Datacenter Performance Analysis of a Tensor Processing Unit," in *Proceedings of the 44th Annual International Symposium on Computer Architecture*, Jun. 2017, pp. 1–12. doi: 10.1145/3079856.3080246.
- [48] O. Jaiyeoba, E. Ogbuju, O. T. Yomi, and F. Oladipo, "Development of a Model to Classify Skin Diseases using Stacking Ensemble Machine Learning Techniques," *J. Comput. Theor. Appl.*, vol. 2, no. 1, pp. 22–38, May 2024, doi: 10.62411/jcta.10488.
- [49] M. Ha, "Top-Heavy CapsNets Based on Spatiotemporal Non-Local for Action Recognition," *J. Comput. Theor. Appl.*, vol. 2, no. 1, pp. 39–50, May 2024, doi: 10.62411/jcta.10551.
- [50] I. Cheheb, "An Investigation into the Impact of Occlusion on Facial Emotion Recognition in the Wild," in *Proceedings of the 17th International Conference on Pervasive Technologies Related to Assistive Environments*, Jun. 2024, pp. 365–368. doi: 10.1145/3652037.3663932.
- [51] D. Ballabio, F. Grisoni, and R. Todeschini, "Multivariate comparison of classification performance measures," *Chemom. Intell. Lab. Syst.*, vol. 174, pp. 33–44, Mar. 2018, doi: 10.1016/j.chemolab.2017.12.004.
- [52] R. E. Ako *et al.*, "Effects of Data Resampling on Predicting Customer Churn via a Comparative Tree-based Random Forest and XGBoost," *J. Comput. Theor. Appl.*, vol. 2, no. 1, pp. 86–101, Jun. 2024, doi: 10.62411/jcta.10562.
- [53] A. N. Safriandono, D. R. I. M. Setiadi, A. Dahlan, F. Z. Rahmanti, I. S. Wibisono, and A. A. Ojugo, "Analyzing Quantum Feature Engineering and Balancing Strategies Effect on Liver Disease Classification," *J. Futur. Artif. Intell. Technol.*, vol. 1, no. 1, pp. 51–63, Jun. 2024, doi: 10.62411/faith.2024-12.
- [54] A. Sikarwar, H. Chandra, and I. Ram, "Real-Time Biometric Verification and Management System Using Face Embeddings," in *2020 IEEE 17th India Council International Conference (INDICON)*, Dec. 2020, pp. 1–4. doi: 10.1109/INDICON49873.2020.9342551.