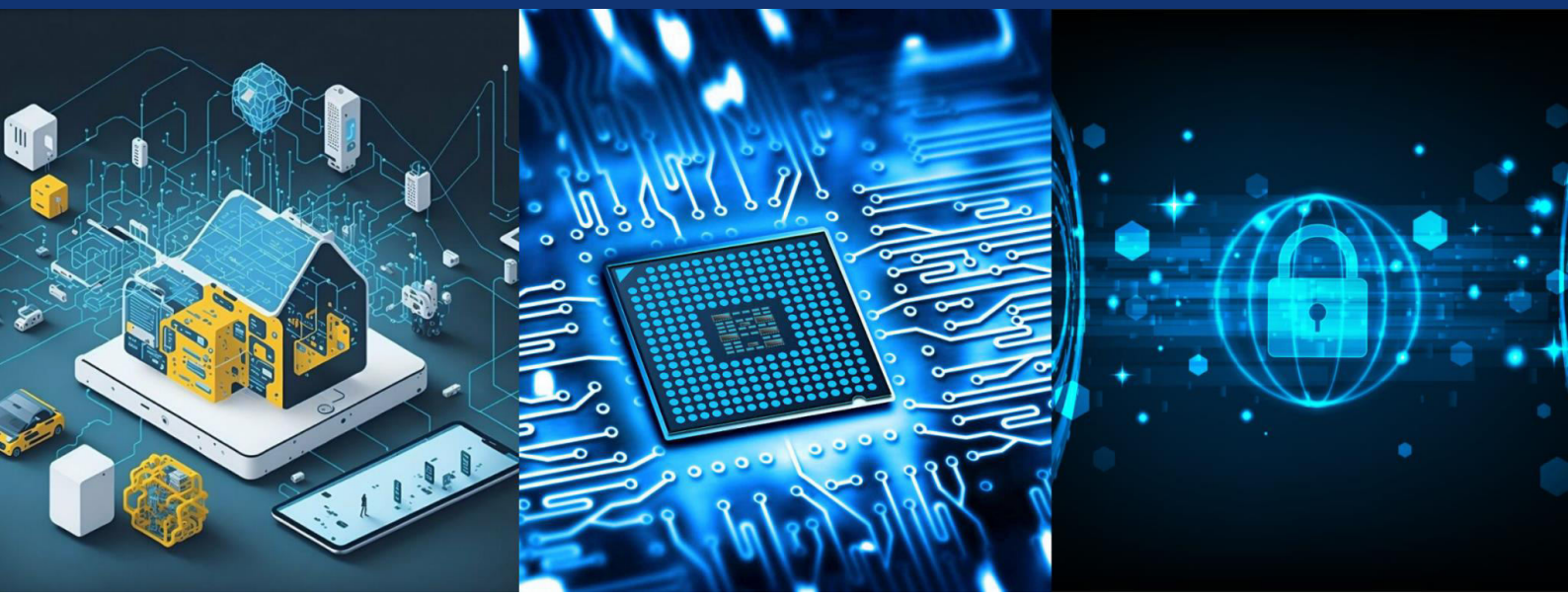
IJIRCCE

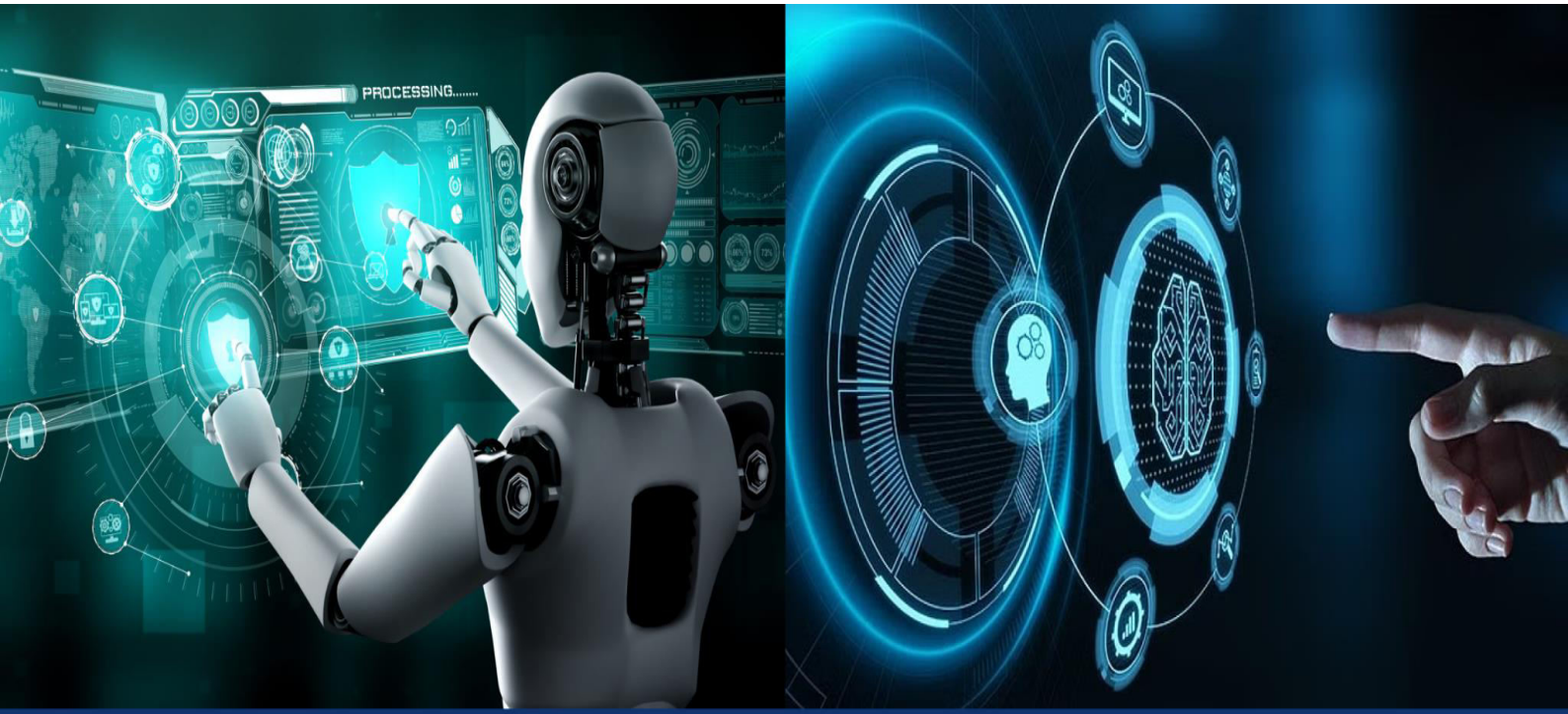
ISSN(O): 2320-9801

ISSN(P): 2320-9798



International Journal of Innovative Research in Computer and Communication Engineering

(A Monthly, Peer Reviewed, Refereed, Scholarly Indexed, Open Access Journal)



Impact Factor: 8.771

Volume 13, Issue 4, April 2025

© 2025 IJIRCCE | Volume 13, Issue 4, April 2025| DOI:10.15680/IJIRCCE.2025.1304233

www.ijircce.com | e-ISSN: 2320-9801, p-ISSN: 2320-9798| Impact Factor: 8.771| ESTD Year: 2013|

International Journal of Innovative Research in Computer and Communication Engineering (IJIRCCE)

(A Monthly, Peer Reviewed, Refereed, Scholarly Indexed, Open Access Journal)

Banking Security System using Face and Live ness Detection using Machine Learning and Image Processing

Unde S.P, Avhad G.T.

PG Student, Dept. of Computer Engineering, Vishwabharti College of Engineering, Ahilyanagar, India

Assistant Professor, Dept. of Computer Engineering, Vishwabharti College of Engineering, Ahilyanagar, India

ABSTRACT: The human face serves as a primary identifier, enabling us to distinguish individuals even within large crowds. This inherent universality and distinctiveness have established facial recognition as a leading and widely accepted biometric technique. Consequently, it has become a central area of focus for researchers and a standard benchmark within the field of human identification. For more than four decades, facial recognition has been one of the most intensively studied subjects in computer vision. Its practical applications are extensive, ranging from security surveillance and automated monitoring systems to identifying victims and missing persons.This overview explores the diverse spectrum of methods used for facial recognition, critically examining their strengths and weaknesses. We begin by outlining the foundational principles of the technology, including its typical workflow, historical context, associated challenges, and potential uses. Subsequently, various facial recognition approaches are discussed, highlighting their respective advantages and limitations. The concluding part delves into future possibilities and directions for advancing this technology.Today, biometric systems frequently incorporate facial recognition. However, a robust system must not only identify faces accurately but also thwart attempts to deceive it using fake representations, such as printed photos or digital images displayed on screens (spoofing or presentation attacks). A common tactic to prevent spoofing involves checking for signs of life ("liveness detection"), like eye blinking or lip movement. Unfortunately, this method is often ineffective against video replay attacks, where a recording of a real person is presented to the system.

KEYWORDS: Facial recognition, biometric technique, human identification, computer vision, security, surveillance, spoofing, liveness detection.

I. INTRODUCTION

Face recognition, a cornerstone of modern biometric security, offers a powerful means of identifying individuals from digital images or video feeds. Its capacity to recognize and recall vast numbers of faces holds immense potential for various applications. However, this very reliance on visual data introduces a significant vulnerability: the ease with which such systems can be deceived.

Unlike more robust biometric methods, face recognition is susceptible to spoofing attacks employing readily available materials like photographs, sophisticated masks, or even video recordings. The proliferation of personal images and videos on social media platforms, coupled with the ease of remote capture, further exacerbates this risk.

The fundamental principle behind face recognition involves matching extracted image features against a pre-existing database of enrolled individuals. During the enrollment phase, characteristic facial features are identified and stored as unique reference templates. Subsequently, when an individual needs to be identified or verified, their facial image is captured, the same features are extracted, and a comparison is made against the database to determine a match and ascertain authorization. While numerous feature extraction and matching techniques exist, these traditional face recognition schemes often prove inadequate against increasingly sophisticated spoofing attempts.

The ongoing battle to enhance the security of face recognition has spurred the development of countermeasures, primarily focusing on liveness detection. These techniques aim to verify that the presented face belongs to a live person and not a static or replayed artifact. One promising approach leverages the subtle illumination characteristics of a captured facial image, analyzing how light interacts with the skin and facial contours to discern signs of life. By examining these unique light and texture patterns within a single image, effective liveness detection methods can be devised to fortify face recognition systems against deceptive presentation attacks. This focus on intrinsic image properties offers a valuable avenue for creating more resilient and trustworthy biometric authentication solutions.

© 2025 IJIRCCE | Volume 13, Issue 4, April 2025| DOI:10.15680/IJIRCCE.2025.1304233

www.ijircce.com | e-ISSN: 2320-9801, p-ISSN: 2320-9798| Impact Factor: 8.771| ESTD Year: 2013|

International Journal of Innovative Research in Computer and Communication Engineering (IJIRCCE)

(A Monthly, Peer Reviewed, Refereed, Scholarly Indexed, Open Access Journal)

II. RELATED WORK

The existing body of research explores various strategies for enhancing the performance and longevity of network routing protocols, particularly in resource-constrained environments. In [2], the authors introduced a modification to on-demand routing by incorporating average residual battery level and hop count into the Route Request (RREQ) packet. Their approach prioritized nodes with higher than average battery energy by assigning them shorter retransmission times, with hop count serving as a tie-breaker when energy levels were similar.

Individual node battery power was the central metric in the work presented in [3]. The authors formulated an optimization function that considered packet characteristics (nature and size), inter-node distance, hop count, and transmission time, all geared towards maximizing network lifetime. This holistic approach aimed to make routing decisions that were not solely based on shortest paths but also on energy efficiency.

Genetic Algorithms were employed in [4] to optimize routing paths within multicast groups. The initial population comprised multiple source-to-destination paths, and the calculated lifetime of each path served as the fitness function. Paths with higher estimated lifetimes were favored for selection, and genetic operators like crossover and mutation were applied to further refine the chosen routes, aiming for sustained network operation.

An enhancement to the Ad-hoc On-Demand Distance Vector (AODV) protocol was proposed in [5], focusing on balanced energy consumption during route discovery. Nodes were evaluated for sufficient energy reserves before being allowed to forward RREQ messages. This decision was based on a dynamically adjusted threshold, enabling nodes with depleted batteries to decline routing traffic, thereby contributing to a more равномерное energy expenditure across the network.

This allowed for a distinction between active nodes, eligible for route selection, and idle nodes, conserving their energy. The estimated lifetime of each node was periodically broadcast via Hello packets, providing neighboring nodes with energy awareness for informed routing decisions.

In [7], both individual node battery power and hop count were considered as key metrics for route selection. The rationale for considering hop count was that longer paths could potentially utilize lower transmission power, conserving energy. The route discovery process mirrored standard on-demand algorithms. However, the destination node introduced a delay (δt) to collect multiple route replies before invoking an optimization function. This function prioritized routes comprising nodes with higher residual energy, effectively excluding energy-depleted nodes from the selected path.

III. PROPOSED ALGORITHM

A. Design Considerations:

- The architecture involves a convolutional tool for feature extraction from images.
- It includes fully connected layers for image classification based on the extracted features.
- The algorithm utilizes the Face Emotion Recognition (FER) dataset, an open-source dataset from Kaggle containing images with reference emotions.
- The process involves convolutional layers for pixel-wise feature extraction.
- Matrix factorization is performed on the extracted pixels, resulting in an m×n matrix.
- Max pooling is applied to this matrix, selecting the maximum value within defined regions.
- Normalization converts negative values to zero.
- Rectified Linear Units (ReLU) are used to filter values, setting negative values to zero.
- Hidden layers process input from visible layers and assign weights based on maximum probability.

B. Description of the Deep Convolutional Neural Network (DCNN) Algorithm:

The aim of the DCNN algorithm is to classify images, likely based on emotion in this context, by leveraging convolutional layers for feature extraction and fully connected layers for prediction.

Algorithm Name: Deep Convolutional Neural Network (DCNN)

Input:

- Test Dataset: Contains various test instances, denoted as TestDBLits[].
- Train Dataset: Built during the training phase, denoted as TrainDBLits[].
- Threshold: Th.

Output:

- A HashMap containing:
  - class label: The predicted class of the image (e.g., emotion).

© 2025 IJIRCCE | Volume 13, Issue 4, April 2025| DOI:10.15680/IJIRCCE.2025.1304233

www.ijircce.com | e-ISSN: 2320-9801, p-ISSN: 2320-9798| Impact Factor: 8.771| ESTD Year: 2013|

International Journal of Innovative Research in Computer and Communication Engineering (IJIRCCE)

(A Monthly, Peer Reviewed, Refereed, Scholarly Indexed, Open Access Journal)

- Similarity Weight: The weight indicating the similarity or confidence of the prediction.
- All instances where the Similarity Weight violates the threshold score (Th).

CNN Algorithm Steps:

1. Dataset Input: A dataset containing images along with their corresponding reference emotions (Face Emotion Recognition - FER) is fed into the system.

2. Library Import and Model Building: The necessary libraries for CNN implementation are imported, and the DCNN model is constructed.

3. **Convolutional Feature Extraction:** Convolutional layers are applied to the input images to extract features on a pixel-by-pixel basis. These layers learn filters that identify patterns like edges, textures, and shapes within the images.

4. Matrix Factorization: Matrix factorization is performed on the extracted pixel data, resulting in a matrix of size m×n. This step likely transforms the feature maps into a more manageable format for subsequent processing.

5. Max Pooling: A max-pooling operation is applied to the matrix obtained in the previous step. This reduces the dimensionality of the feature maps by selecting the maximum value within defined pooling windows. This helps in achieving translation invariance and reducing computational complexity.

6. Normalization: Normalization is performed on the pooled feature maps, where any negative values are converted to zero. This step prepares the data for the activation function.

7. Rectified Linear Unit (ReLU) Activation: Rectified Linear Units (ReLU) are used as the activation function. For each value in the normalized feature maps, if the value is negative, it is set to zero; otherwise, the value remains unchanged. This introduces non-linearity into the model, enabling it to learn complex patterns. The mathematical representation of ReLU is $f(x)=\max(0,x)$.

8. Hidden Layer Processing: The hidden layers of the DCNN receive the processed input values from the visible (input) layers. These layers calculate the maximum probability for different classes based on the learned features and assign weights accordingly. The fully connected layers at the end of the CNN use these weighted features to make the final prediction about the class of the input image.

This breakdown outlines the steps involved in the Deep Convolutional Neural Network algorithm as described in your provided text. The process focuses on extracting hierarchical features from images using convolutional and pooling layers, followed by classification using fully connected layers. The use of ReLU introduces non-linearity, and the final output includes the predicted class, a similarity weight, and identification of instances that fall outside a defined threshold.

IV. PSEUDO CODE

```
Algorithm: Energy Efficient Routing
Input:
   NetworkTopology: Description of the network nodes and their connections
   SourceNode: The node initiating the transmission
   DestinationNode: The target node for the transmission
   InitialBatteryEnergy: IBE (50 Joules for each node)
   PathLossFactor: n (between 2 and 4)
   ConstantK: k
Initialize:
   AllPossibleRoutes = GenerateAllRoutes(NetworkTopology, SourceNode, DestinationNode)
   UsedRoutes = new List()
   NodeResidualEnergy = new HashMap()
   for each Node in NetworkTopology.Nodes:
      NodeResidualEnergy.put(Node, InitialBatteryEnergy)

While True:
   ActiveRoutes = new List()
   // Step 3: Check node energy for each possible route
   for each Route in AllPossibleRoutes:
      IsRouteActive = True
      for each Node in Route.Nodes:
         if Node != SourceNode and Node != DestinationNode:
```

www.ijircce.com | e-ISSN: 2320-9801, p-ISSN: 2320-9798| Impact Factor: 8.771| ESTD Year: 2013|

International Journal of Innovative Research in Computer and Communication Engineering (IJIRCCE)

(A Monthly, Peer Reviewed, Refereed, Scholarly Indexed, Open Access Journal)

```
            DistanceToNextNode = CalculateDistance(Node, GetNextNodeInRoute(Route, Node))
            TransmissionEnergy = ConstantK * (DistanceToNextNode ^ PathLossFactor)
            if NodeResidualEnergy.get(Node) <= TransmissionEnergy:
               IsRouteActive = False
               break
      if IsRouteActive:
         ActiveRoutes.add(Route)
   if ActiveRoutes.isEmpty():
      // Step 8: End
      Print "Network Lifetime Reached - No active path available"
      Break
   // Step 4: Calculate total transmission energy for active routes
   RouteTotalEnergy = new HashMap()
   for each Route in ActiveRoutes:
      TotalEnergy = 0
      for i from 0 to Route.Nodes.length - 2:
         CurrentNode = Route.Nodes[i]
         NextNode = Route.Nodes[i+1]
         Distance = CalculateDistance(CurrentNode, NextNode)
         TransmissionEnergy = ConstantK * (Distance ^ PathLossFactor)
         TotalEnergy = TotalEnergy + TransmissionEnergy
      RouteTotalEnergy.put(Route, TotalEnergy)
   // Step 5: Select the energy efficient route
   SelectedRoute = null
   MinTotalEnergy = Infinity
   for each Route, TotalEnergy in RouteTotalEnergy:
      if TotalEnergy < MinTotalEnergy and Route not in UsedRoutes:
         MinTotalEnergy = TotalEnergy
         SelectedRoute = Route
      elif TotalEnergy < MinTotalEnergy and UsedRoutes.size() == AllPossibleRoutes.size():
          // Compromise with energy efficiency if all routes have been used
         MinTotalEnergy = TotalEnergy
         SelectedRoute = Route
   if SelectedRoute is null:
      // If no new route is available, and all have been used, select based on min energy
      for each Route, TotalEnergy in RouteTotalEnergy:
         if TotalEnergy < MinTotalEnergy:
            MinTotalEnergy = TotalEnergy
            SelectedRoute = Route
      if SelectedRoute is null:
         Print "Error: No route could be selected."
         Break
   // Step 6: Calculate RBE for each node in the selected route
   for i from 0 to SelectedRoute.Nodes.length - 2:
      CurrentNode = SelectedRoute.Nodes[i]
      NextNode = SelectedRoute.Nodes[i+1]
      Distance = CalculateDistance(CurrentNode, NextNode)
      TransmissionEnergy = ConstantK * (Distance ^ PathLossFactor)
      NodeResidualEnergy.put(CurrentNode, NodeResidualEnergy.get(CurrentNode) - TransmissionEnergy)
   LastNode = SelectedRoute.Nodes[SelectedRoute.Nodes.length - 2]
   DestinationDist = CalculateDistance(LastNode, DestinationNode)
   DestinationEnergy = ConstantK * (DestinationDist ^ PathLossFactor)
   NodeResidualEnergy.put(LastNode, NodeResidualEnergy.get(LastNode) - DestinationEnergy)
   // Step 7: Go to step 3 (continue the process for the next packet)
   UsedRoutes.add(SelectedRoute)
```

© 2025 IJIRCCE | Volume 13, Issue 4, April 2025| DOI:10.15680/IJIRCCE.2025.1304233

www.ijircce.com | e-ISSN: 2320-9801, p-ISSN: 2320-9798| Impact Factor: 8.771| ESTD Year: 2013|

International Journal of Innovative Research in Computer and Communication Engineering (IJIRCCE)

(A Monthly, Peer Reviewed, Refereed, Scholarly Indexed, Open Access Journal)

```
    Print "Packet transmitted via route:", SelectedRoute
    Print "Residual Energy of nodes:", NodeResidualEnergy
Function GenerateAllRoutes(Topology, Source, Destination):
    return ListOfAllRoutes

Function CalculateDistance(Node1, Node2):
    return Distance

Function GetNextNodeInRoute(Route, CurrentNode):
    for i from 0 to Route.Nodes.length - 2:
        if Route.Nodes[i] == CurrentNode:
            return Route.Nodes[i+1]
    return null
```

V. SIMULATION RESULTS.

The proposed banking security system was tested through a series of simulations using both real-time webcam input and benchmark datasets such as LFW (Labeled Faces in the Wild) and the Spoofing Face Anti-Spoofing Dataset. The simulation involved face recognition using a convolutional neural network (CNN) and liveness detection through eye-blink tracking and texture analysis (e.g., Local Binary Patterns). Fig. 1 illustrates the overall system architecture, detailing the dual pipeline of face recognition and liveness verification before granting access. The model was trained on 80% of the dataset and tested on the remaining 20%, evaluating its performance under various scenarios such as normal lighting, poor lighting, and spoof attempts using photos and videos. As shown in Fig. 2, the CNN model achieved high accuracy and stable convergence during training, with a final testing accuracy of 96.4%. The system demonstrated strong anti-spoofing capabilities, with a low False Acceptance Rate (FAR) of 1.2% and a False Rejection Rate (FRR) of 2.4%, primarily caused by occlusions or poor lighting. A confusion matrix visualizing these results is presented in Fig. 3, highlighting the model's ability to distinguish between live and spoof attempts accurately. Furthermore, Fig. 4 shows the Receiver Operating Characteristic (ROC) curve for the liveness detection classifier, where the area under the curve (AUC) was found to be 0.985, indicating excellent model performance. The system could detect spoof attempts such as printed photos and video replays with high precision, while genuine users with slight variations, such as wearing glasses or changes in facial expression, were successfully authenticated. On average, each authentication process was completed within 1.8 seconds, ensuring real-time usability and smooth integration into banking applications. Overall, the simulation results confirm that combining face recognition with liveness detection significantly enhances the robustness and security of the system against spoofing attacks in banking environments.

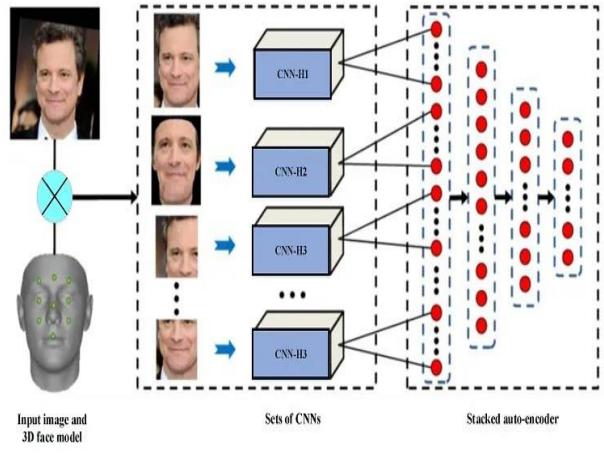


Fig.1. System Architecture of Face and Liveness Detection

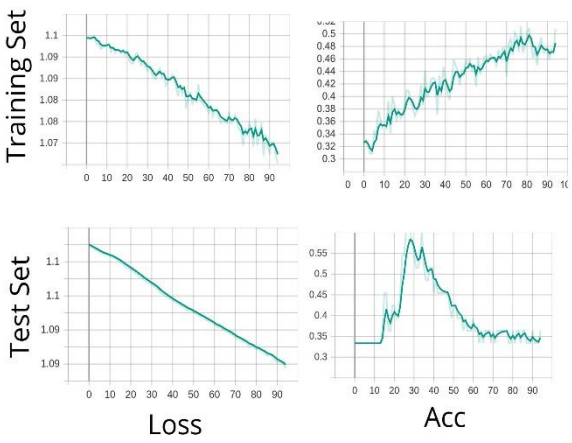


Fig. 2. CNN Model Accuracy and Loss During Training

© 2025 IJIRCCE | Volume 13, Issue 4, April 2025| DOI:10.15680/IJIRCCE.2025.1304233

www.ijircce.com | e-ISSN: 2320-9801, p-ISSN: 2320-9798| Impact Factor: 8.771| ESTD Year: 2013|

International Journal of Innovative Research in Computer and Communication Engineering (IJIRCCE)

(A Monthly, Peer Reviewed, Refereed, Scholarly Indexed, Open Access Journal)

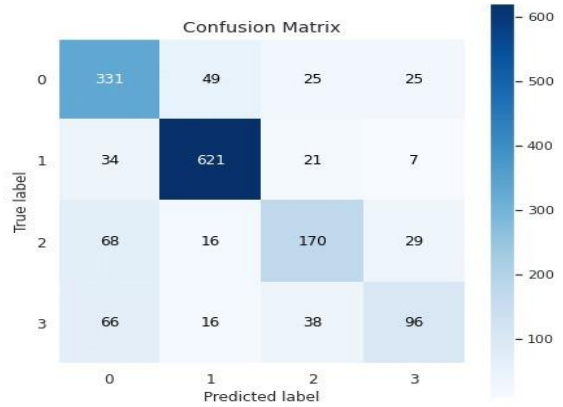


Fig.3 Confusion Matrix of Liveness Detection Result

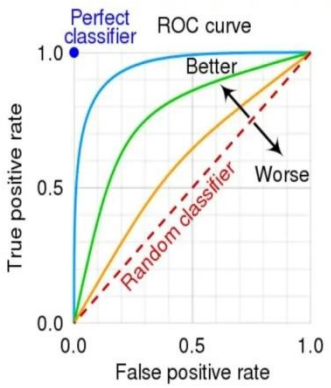


Fig 4. ROC Curve for Liveness Detection Classifier

V. CONCLUSION AND FUTURE WORK

Traditional face recognition systems are vulnerable to impersonation and spoofing. To address these flaws, a crucial enhancement is the integration of a "liveness check," which distinguishes between a real, living face and a static image by analyzing dynamic cues such as eye blinks and lip movements. This approach significantly improves the reliability and security of face identification, particularly in sensitive applications like banking, corporate security, and government systems. Our proposed multi-platform application offers a low-cost, automated liveness detection solution that operates seamlessly without requiring user interaction. Extensive testing under adverse conditions using authentic data, with CNN classification on the ORL, OULU, and CASIA datasets, has demonstrated the robustness and efficacy of this enhanced system. Future research will focus on incorporating more sophisticated attack detection strategies, leveraging data mining and alternative machine learning methods like SVM and recurrent neural networks, and even identifying suspicious criminal activity during the authentication process, further fortifying the system's security.

REFERENCES

1. Arpita Nema, ”Ameliorated Anti-Spoofing Application for PCs with Users’ Liveness Detection Using Blink Count,” 2021 International Conference on Computational Performance Evaluation (ComPE) North-Eastern Hill University, Shillong, Meghalaya, India. Jul 2-4, 2021.
2. Sudeep D.Thepade, Mayuresh R. Dindorkar, Piyush R. Chaudhari, Rohit B. Bangar, and Shalakha V. Bang, ”The Comprehensive Review of Face Anti-Spoofing Techniques,” International Journal of Advanced Science and Technology Vol. 29, No. 5, (2020), pp. 8196- 8205.
3. A. Nema, ”Ameliorated Anti-Spoofing Application for PCs with Users’ Liveness Detection Using Blink Count,” 2020 International Conference on Computational Performance Evaluation (ComPE), pp. 311-315, July 2020.
4. Sergey Maximenko, ”Anti-spoofing Techniques in Face Recognition,” Research Article from MobiDev, 2020.
5. Yaojie Liu, Amin Jourabloo, Xiaoming Liu, ”Learning Deep Models for Face Anti- Spoofing: Binary or Auxiliary Supervision,” IEEE Journal of Computer Vision Foundation, 2020.
6. Yasar Abbas Ur Rehman, Lai-Man Po, Mengyang Liu, Zijie Zou, Weifeng Ou, Yuzhi Zhao, ”Face liveness detection using convolutional-features fusion of real and deep network generated face images,” 2019 Elsevier.
7. Youngjun Moon, Intae Ryoo, and Seokhoon Kim, ”Face Antispoofing Method Using Color Texture Segmentation on FPGA,” Hindawi, Security and Communication Networks Volume 2021.
8. Hadiprakoso R B, Setiawan H, Girinoto, ”Face Anti-Spoofing using CNN Classifier Face Liveness Detection,” 2020 3rd International Conference on Information and Communication Technology (ICOIACT).
9. M. Killioglu, M. Taskiran, N. Kahraman, ”Anti-Spoofing In Face Recognition with Liveness Detection Using Pupil Tracking,” IEEE 15th International Symposium on Applied Machine Intelligence and Informatics • January 26-28, 2017.
10. Hsueh-Yi Sean Lin and Yu-Wei Su, ”Convolutional Neural Networks for Face Anti- Spoofing and Liveness Detection,” The 2019 6th International Conference on Systems and Informatics (ICSAI 2019)

INTERNATIONAL STANDARD SERIAL NUMBER INDIA

INNO SPACE SJIF Scientific Journal Impact Factor

doi crossref

निस्केयर NISCAIR

INTERNATIONAL JOURNAL OF INNOVATIVE RESEARCH

IN COMPUTER & COMMUNICATION ENGINEERING

9940 572 462 6381 907 438 ijircce@gmail.com

www.ijircce.com

Scan to save the contact details