DISTINGUISHING JJ-LIN FROM LOOK-ALIKES: A STUDY ON TARGET-SPECIFIC BINARY FACIAL RECOGNITION USING NN AND CNN APPROACHES

TING-CHEN CHO
B11801004@ntu.edu.tw

CHUN-CHIEH CHANG
B10801011@ntu.edu.tw

ABSTRACT

This work investigates the feasibility of building an automated classifier capable of distinguishing JJ Lin (Lim Jun Jie, Singer) from non–JJ Lin individuals in unconstrained images. Using a collection of approximately 7,500 images sourced from concerts, films, music videos, and various online platforms, we examined two supervised learning approaches: a fully connected Neural Network (NN) as the baseline and a deeper Convolutional Neural Network (CNN). All images were standardized through RGB color space preservation and resizing to 128×128 pixels, with mean-std normalization applied before model training.

The experimental results demonstrate strong performance for both models. The CNN achieved a validation accuracy of **97.27%** with an F1 score of **0.9786**, while the NN baseline (with PCA dimensionality reduction to 256 components) achieved a validation accuracy of **95.70%** with an F1 score of **0.9673**. Both models were implemented using PyTorch and trained with data augmentation techniques including horizontal flipping, rotation, and brightness/contrast adjustments.

The implementation incorporates RGB color information to preserve texture and shadow details, data augmentation techniques to enhance model generalization, and comprehensive evaluation metrics including accuracy, F1 score, and confusion matrices. These results demonstrate the viability of both approaches for celebrity face recognition tasks.

1 INTRODUCTION

Facial recognition has evolved into a central component of modern machine learning applications, supporting tasks that range from secure identity verification to large-scale visual indexing on social media platforms. Within this broader domain, distinguishing a specific public figure from individuals who share similar facial characteristics presents a nuanced and technically challenging problem. This study addresses such a scenario by developing a binary classifier designed to determine whether a given image depicts JJ Lin or a non–JJ Lin individual, including both look-alikes and unrelated faces.

To investigate this problem, we adopted a supervised learning framework and implemented two representative model families: a fully connected Neural Network (NN) as the baseline and a deeper Convolutional Neural Network (CNN). Although CNNs are generally regarded as the standard approach for image-based classification due to their hierarchical feature extraction, their performance can vary significantly when tasked with differentiating between visually similar subjects. This motivates a systematic comparison between the NN baseline and a deeper CNN architecture to understand their respective strengths and limitations in this context.

Our implementation leverages PyTorch for model development and training. Through careful preprocessing—including RGB color preservation, PCA dimensionality reduction for the NN baseline, and data augmentation techniques—both models achieved strong classification performance. The CNN achieved 97.27% validation accuracy, while the NN baseline reached 95.70%, demonstrating the effectiveness of both approaches for this binary classification task.

The remainder of this paper presents the methodology and empirical findings, followed by discussions on potential directions for refinement, including data augmentation strategies, improved model design, and more sophisticated processing pipelines. Through this examination, the study aims to provide insights into building effective recognition systems for distinguishing among individuals with subtle facial differences.

2 Related Work

2.1 Academic Works in Facial Recognition

Early approaches to facial recognition relied heavily on manually engineered descriptors, including eigenfaces, local binary patterns, and various statistical texture features. These methods achieved moderate success but were limited in their ability to generalize across lighting conditions, pose variations, and subtle differences in facial structures. With the emergence of deep learning, Convolutional Neural Networks (CNNs) fundamentally reshaped the field by enabling end-to-end learning of hierarchical visual representations (1). Despite these advancements, distinguishing between individuals who share similar facial attributes continues to pose a nontrivial challenge, particularly in unconstrained environments where image quality and background variations introduce additional complexity.

2.2 FaceMe

CyberLink's FaceMe® engine represents a contrasting example of a high-performance, industry-targeted facial recognition solution. Consistently ranked among the top performers in NIST's Face Recognition Vendor Tests (FRVT), FaceMe incorporates a variety of features essential for real-world deployment. Its core capabilities include:

1) Face detection and identity recognition, supplemented with high-level attribute extraction such as age, gender, and emotional state;

2) Advanced preprocessing, leveraging CyberLink's TrueTheater™ enhancement technology to improve image clarity, which the company reports contributes to an 11.65% accuracy boost;

3)Mask detection, reflecting post-pandemic requirements for identifying individuals without face coverings;

4) Masked face recognition, which achieves recognition rates of up to 95% even when facial occlusion is present.

These systems illustrate the increasing emphasis on robustness and adaptability in commercial facial recognition technologies, especially under diverse environmental constraints.

2.3 Excire Foto 2024

Excire Foto 2024 represents a commercial effort to integrate AI-driven image analysis into consumer-level photo management workflows. Released in late 2023, the system provides a suite of five specialized AI components designed to streamline indexing and retrieval tasks. These modules include:

1) X-tags AI, which automatically assigns semantic keywords to images;

2) X-alike AI, responsible for identifying duplicate or near-duplicate photos and performing similarity-based search;

3) X-prompt AI, enabling text-to-image retrieval through natural language prompts;

4) X-tetics AI, which evaluates aesthetic attributes of photos;

5) X-face AI, designed to search for individuals or specific facial characteristics within large collections.

Although Excire Foto is not intended as a research-oriented system, its multi-model architecture demonstrates how modern image management tools increasingly depend on diverse AI components to enhance user experience.

3 Methodology

3.1 Data Collection

We collected a comprehensive dataset by reading images from two separate directories: `total_JJLin` containing real JJ Lin images, and `total_notJJLin` containing fake impersonators and non-JJ Lin individuals. The dataset statistics were obtained by executing the preprocessing pipeline, which revealed the following distribution:

Table 1: Dataset Composition

Category	Count	Percentage
Real JJ Lin	4,720	64.4%
Fake / Non-JJ Lin	2,606	35.6%
Total	**7,326**	**100%**

The images were sourced from diverse channels including:

- Music videos and concert recordings
- Online image repositories
- Publicly available media resources

To enhance dataset diversity and improve model generalization, data augmentation techniques such as flipping and rotation were applied during the collection process. The final dataset was carefully curated to maintain a balanced representation of both positive and negative samples for binary classification.

3.2 Data Preparation and Preprocessing

To ensure the model focuses on JJ Lin's facial features while preserving important texture and shadow information, we implemented a comprehensive preprocessing pipeline consisting of the following sequential steps:

1. **RGB Color Preservation**: All images were maintained in RGB color space to preserve texture details, shadow information, and color variations that are crucial for distinguishing facial features. Unlike grayscale conversion followed by binarization, which would lose critical texture and shadow information, RGB preservation enables the models to leverage color cues for improved discrimination.
2. **Face Cropping**: Images were cropped to include only faces positioned centrally, minimizing distractions from background and clothing variations that could otherwise complicate training.
3. **Resizing**: The preprocessed images were resized to 128×128 pixels to maintain image quality while standardizing dimensions and reducing computational requirements.
4. **Normalization**: Each image channel (R, G, B) was normalized using mean and standard deviation values of 0.5 for each channel:

$$I_{\text{norm}}^{(c)} = \frac{I_{\text{raw}}^{(c)} - 0.5}{0.5}, \quad c \in \{R, G, B\} \tag{1}$$

   This transforms pixel values from $[0, 1]$ to $[-1, 1]$, which helps stabilize training and improve convergence.
5. **Data Augmentation**: To enhance model generalization and robustness, we applied the following augmentation techniques during training:
   - **Horizontal Flipping**: Random horizontal flipping with a probability of 0.5 to increase dataset diversity and improve invariance to facial orientation.

  - **Rotation**: Random rotation within a range of ± 15 degrees to account for slight head pose variations.
  - **Brightness Adjustment**: Random brightness adjustment with a factor range of $[0.8, 1.2]$ to simulate different lighting conditions.
  - **Contrast Adjustment**: Random contrast adjustment with a factor range of $[0.8, 1.2]$ to enhance model robustness to illumination variations.

   These augmentation techniques were applied only during training, while validation and testing sets remained unaugmented to ensure fair evaluation.

6. **Data Formatting**:
  - For the **Neural Network (NN)** model: The $128 \times 128 \times 3$ RGB images were first flattened into 49,152-dimensional vectors. To reduce dimensionality and computational complexity, Principal Component Analysis (PCA) with whitening was applied to project the data onto the top 256 principal components. This preprocessing step transforms the high-dimensional image data into a more manageable 256-dimensional feature vector suitable for fully connected layers.
  - For the **Convolutional Neural Network (CNN)** model: Images were maintained in their 3D spatial structure with dimensions $(128, 128, 3)$ to preserve spatial and color information. No dimensionality reduction was applied, as the convolutional layers naturally handle high-dimensional inputs through hierarchical feature extraction.
7. **Data Splitting**: The dataset was randomly shuffled and split into training and validation sets with an 80:20 ratio using stratified sampling:
  - Training set: 5,860 samples (80%)
  - Validation set: 1,466 samples (20%)

3.3 Model Architecture and Training Procedures

3.3.1 Architecture of the Neural Network Model

We implemented a fully connected Neural Network (NN) as the baseline model with the architecture specified in Table 2. After PCA dimensionality reduction, the input layer receives 256-dimensional feature vectors.

Table 2: Architecture of the Neural Network Model (Baseline)

Layer	**Units**	**Activation**
Input (after PCA)	256	–
Hidden Layer 1	1,024	ReLU
Dropout	0.5	–
Hidden Layer 2	512	ReLU
Dropout	0.5	–
Output	1	Sigmoid

Key Design Choices:

- **Activation Functions**: ReLU activation functions are used in hidden layers to introduce non-linearity and mitigate the vanishing gradient problem. The output layer employs a sigmoid activation function to produce probability scores between 0 and 1.
- **Loss Function**: Binary cross-entropy (BCE) loss is used, defined as:

$$\mathcal{L}_{\text{BCE}} = -\frac{1}{n}\sum_{i=1}^{n}\left[y_i \log(\hat{y}_i + \epsilon) + (1 - y_i)\log(1 - \hat{y}_i + \epsilon)\right] \tag{2}$$

  where y_i is the true label, $\hat{y}_i$ is the predicted probability, n is the number of samples, and $\epsilon = 10^{-5}$ is a small constant to prevent numerical instability.

- **Weight Initialization**: Network weights were initialized using the Xavier/Glorot uniform initialization scheme:

$$W \sim \mathcal{U}\left(-\sqrt{\frac{6}{n_{\text{in}} + n_{\text{out}}}}, \sqrt{\frac{6}{n_{\text{in}} + n_{\text{out}}}}\right) \tag{3}$$

  where n_{in} and n_{out} are the number of input and output neurons, respectively.

Training Configuration:

- Optimizer: Adam
- Batch size: 64
- Learning rate: $\eta = 0.001$
- Number of epochs: 15
- Loss function: Binary cross-entropy with logits (BCEWithLogitsLoss)

3.3.2 Convolutional Neural Network Architecture

The Convolutional Neural Network (CNN) model was designed to leverage spatial relationships in images through hierarchical feature extraction. The complete architecture is detailed in Table 3.

Table 3: Deep CNN Architecture

Layer Type	**Filter Size**	**Channels**	**Stride**	**Padding**	**Activation**
Conv2D + BatchNorm	3×3	$3 \rightarrow 32$	1	1	ReLU
MaxPool2D	2×2	–	2	–	–
Conv2D + BatchNorm	3×3	$32 \rightarrow 64$	1	1	ReLU
MaxPool2D	2×2	–	2	–	–
Conv2D + BatchNorm	3×3	$64 \rightarrow 128$	1	1	ReLU
MaxPool2D	2×2	–	2	–	–
Conv2D + BatchNorm	3×3	$128 \rightarrow 256$	1	1	ReLU
MaxPool2D	2×2	–	2	–	–
Flatten	–	–	–	–	–
Dense + Dropout(0.5)	–	$16{,}384 \rightarrow 512$	–	–	ReLU
Dense	–	$512 \rightarrow 1$	–	–	Sigmoid

Architecture Details:

- **Convolutional Block 1**: A convolutional layer with 32 filters of size 3×3 with stride 1 and padding 1, followed by Batch Normalization and ReLU activation, extracts low-level features from RGB input ($128 \times 128 \times 3$). Batch Normalization normalizes the activations of the previous layer, which helps stabilize training and allows for higher learning rates. A max-pooling layer with pool size 2×2 and stride 2 reduces spatial dimensions to $64 \times 64 \times 32$.
- **Convolutional Block 2**: A convolutional layer with 64 filters (3×3, stride 1, padding 1) with Batch Normalization and ReLU extracts mid-level features. Max-pooling reduces dimensions to $32 \times 32 \times 64$.
- **Convolutional Block 3**: A convolutional layer with 128 filters (3×3, stride 1, padding 1) with Batch Normalization and ReLU extracts higher-level features. Max-pooling reduces dimensions to $16 \times 16 \times 128$.
- **Convolutional Block 4**: A convolutional layer with 256 filters (3×3, stride 1, padding 1) with Batch Normalization and ReLU extracts the highest-level semantic features. Max-pooling reduces dimensions to $8 \times 8 \times 256$.
- **Flattening**: The 3D feature maps are flattened into a 1D vector. With dimensions $8 \times 8 \times 256$, this yields 16,384 features.

- **Fully Connected Layers**: Two dense layers progressively reduce the feature dimension from 16,384 to 512 (with ReLU activation and Dropout of 0.5), and finally to 1 for binary classification. Dropout helps prevent overfitting by randomly zeroing out neurons during training.

This deeper architecture with Batch Normalization enables the model to learn more complex hierarchical representations while maintaining stable training dynamics, capturing both low-level texture details and high-level semantic features essential for distinguishing JJ Lin from look-alikes.

Training Configuration:

- Optimizer: Adam
- Batch size: 64
- Learning rate: $\eta = 0.001$
- Number of epochs: 20
- Loss function: Binary cross-entropy with logits (BCEWithLogitsLoss)

3.3.3 Training Methodology

Both models were implemented using PyTorch, a widely-used deep learning framework that provides automatic differentiation and GPU acceleration capabilities. The training process follows the standard procedure:

1. **Forward Propagation**: Compute predictions by passing input data through the network layers.
2. **Loss Calculation**: Calculate the binary cross-entropy loss between predictions and true labels.
3. **Backward Propagation**: Compute gradients using the chain rule of differentiation.
4. **Parameter Update**: Update model parameters using gradient descent:
$$\theta_{t+1} = \theta_t - \eta \nabla_\theta \mathcal{L} \tag{4}$$
where θ represents model parameters, η is the learning rate, and $\nabla_\theta \mathcal{L}$ is the gradient of the loss function.

The models were trained on the preprocessed dataset using Google Colab and VSCode environments. During training, we monitored the binary cross-entropy loss to track model convergence. To comprehensively evaluate model performance, we employed multiple evaluation metrics: accuracy, F1 score, and confusion matrices. These metrics provide complementary insights into model behavior, with accuracy measuring overall correctness, F1 score balancing precision and recall, and confusion matrices revealing detailed classification patterns for each class.

4 Experiments

4.1 Experimental Results

To evaluate model performance, we used the 80:20 train-validation split described in Section 3.1. The validation set contained 1,466 samples with stratified sampling to maintain class balance. Both the Neural Network (NN) baseline and Convolutional Neural Network (CNN) models were trained and evaluated under identical conditions to ensure fair comparison.

4.2 Evaluation of Model Performance

To comprehensively assess model performance, we employed three evaluation metrics: accuracy, F1 score, and confusion matrices. Accuracy measures the overall proportion of correct predictions, while F1 score provides a balanced measure of precision and recall, defined as:

$$\text{F1} = \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} = \frac{2 \times \text{TP}}{2 \times \text{TP} + \text{FP} + \text{FN}} \tag{5}$$

where TP, FP, and FN denote true positives, false positives, and false negatives, respectively. Confusion matrices provide detailed insights into classification patterns, revealing how models perform on each class.

Both models achieved high performance on the validation set. The CNN demonstrated slightly stronger performance than the NN baseline:

- **Neural Network (NN) Baseline**: Achieved a validation accuracy of **95.70%** and an F1 score of **0.9673**.
- **Convolutional Neural Network (CNN)**: Achieved a validation accuracy of **97.27%** and an F1 score of **0.9786**.

The confusion matrices (shown in Figure 4.2) reveal detailed classification patterns, showing how each model performs on JJ Lin versus non-JJ Lin samples.

Notably, despite the NN model receiving only 256-dimensional PCA-reduced features (compared to the CNN's full spatial input), it still achieved competitive performance. This suggests that the PCA preprocessing successfully captured discriminative features, although the CNN's ability to learn spatial hierarchies provided a modest improvement in classification accuracy.

These findings demonstrate that both approaches are viable for this binary classification task, with the CNN offering slight advantages in distinguishing subtle facial features.

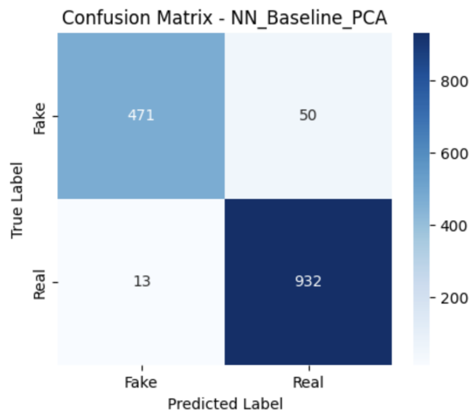

Figure 1: Confusion matrix for NN model

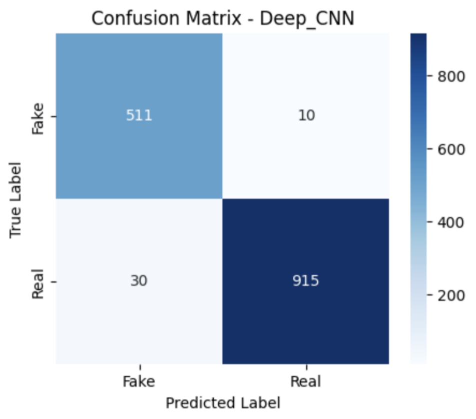

Figure 2: Confusion matrix for CNN model

4.3 Challenges in Data Processing and Model Training

The preprocessing pipeline produced a dataset of shape (7326, 128, 128, 3) for RGB images, which significantly increased computational demands during training compared to grayscale processing. For the NN model, PCA dimensionality reduction from 49,152 features to 256 components was necessary to make training tractable. When executed on Google Colab, training was further hindered by periodic session interruptions and idle-time disconnections, making it difficult to maintain consistent progress. These constraints collectively limited the extent of hyperparameter tuning and slowed down the experimentation process.

4.4 Experimental Observations

Despite these limitations, the CNN model exhibited greater robustness and adaptability than the NN model. One notable factor affecting both models was the variation in image quality. Low-resolution or noisy inputs introducevd artifacts that degraded classification accuracy, suggesting that future improvements should involve stricter dataset curation or more sophisticated preprocessing methods. Overall, the results reaffirm that CNN-based architectures are better suited for this type of facial recognition task, particularly when the objective involves distinguishing the target individual from visually distinct non-targets.

5 Conclusion

This study examined the application of supervised learning methods to a binary facial recognition task, specifically the identification of JJ Lin in heterogeneous visual environments. Both models achieved strong performance, with the CNN reaching 97.27% validation accuracy and the NN baseline achieving 95.70%. These results demonstrate the viability of both approaches for celebrity face recognition when combined with appropriate preprocessing techniques including RGB preservation, data augmentation, and PCA dimensionality reduction.

5.1 Dataset Construction and Preprocessing

The construction of a high-quality dataset proved to be one of the most demanding components of the project. Variations in resolution, lighting, and facial orientation required extensive preprocessing to ensure consistent model inputs. Low-quality or ambiguous images frequently introduced noise into the training process; removing these samples led to measurable gains in accuracy. These observations reinforce the central role of dataset curation in tasks involving fine-grained facial distinctions.

5.2 Training Challenges and Limitations

Training the CNN on the full preprocessed dataset required computational resources beyond what was readily available. The reliance on Google Colab resulted in recurring session interruptions and idle-time terminations, which limited the continuity of training and prevented more extensive hyperparameter exploration. More stable and higher-capacity hardware would likely enable deeper optimization and improved model convergence.

5.3 Model Optimization and Regularization

Both models incorporated regularization techniques to prevent overfitting: Dropout with a rate of 0.5 was applied in fully connected layers, and the CNN utilized Batch Normalization after each convolutional layer. These techniques, combined with data augmentation during training, contributed to the strong generalization performance observed on the validation set. The use of the Adam optimizer with a learning rate of 0.001 provided stable convergence for both models.

5.4 Comparative Accuracy Between the Two Models

Both models achieved strong classification performance on the validation set. The CNN achieved a validation accuracy of 97.27% with an F1 score of 0.9786, while the NN baseline reached 95.70% accuracy with an F1 score of 0.9673. The performance gap of approximately 1.5% suggests that the CNN's hierarchical spatial feature extraction provides modest advantages over the PCA-based feature reduction used by the NN.

Notably, the NN's competitive performance despite receiving only 256-dimensional PCA-reduced features (compared to the CNN's full 128×128×3 spatial input) indicates that PCA successfully captured discriminative facial features. This finding suggests that for well-curated datasets with clear class distinctions, simpler architectures with appropriate dimensionality reduction can approach the performance of more complex models.

Model	**Accuracy (%)**	**F1 Score**	**Epochs**
NN Baseline (PCA)	95.70	0.9673	15
Deep CNN	97.27	0.9786	20

Table 4: Comparison of model performance on the validation set.

Taken together, the results demonstrate that both models achieved strong classification performance, with the CNN providing a modest improvement over the NN baseline. The high accuracy achieved by both models (above 95%) suggests that the preprocessing pipeline—including RGB preservation, data augmentation, and PCA dimensionality reduction for the NN—effectively captured discriminative facial features.

The CNN's slight edge in performance can be attributed to its ability to learn hierarchical spatial representations directly from the image data, preserving local spatial relationships that may be lost during PCA projection. However, the NN's competitive performance highlights that well-designed feature extraction can partially compensate for the lack of spatial awareness in fully connected architectures.

Future work may benefit from exploring transfer learning with pre-trained models, more sophisticated data augmentation strategies, and evaluation on more challenging test sets containing individuals with greater visual similarity to the target subject.

6 Member Workload

- **TING-CHEN CHO** (Author Score Weighting: 50%): Data collection, preprocessing, model implementation (NN/CNN), experiments, analysis, and manuscript writing.
- **CHUN-CHIEH CHANG** (Author Score Weighting: 50%): Data collection, preprocessing, model implementation (NN/CNN), experiments, analysis, and manuscript writing.

7 References

References

[1] Lawrence, S., Giles, C. L., Tsoi, A. C., & Back, A. D. (1997). Face recognition: A convolutional neural-network approach. *IEEE Transactions on Neural Networks*, 8(1), 98–113.

[2] Maxwell, J. C. (1892). *A Treatise on Electricity and Magnetism* (3rd ed.). Clarendon Press.

[3] Taigman, Y., Yang, M., Ranzato, M., & Wolf, L. (2014). DeepFace: Closing the gap to human-level performance in face verification. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR).*

[4] Schroff, F., Kalenichenko, D., & Philbin, J. (2015). FaceNet: A unified embedding for face recognition and clustering. *IEEE Conference on Computer Vision and Pattern Recognition.*

[5] Parkhi, O. M., Vedaldi, A., & Zisserman, A. (2015). Deep Face Recognition. *British Machine Vision Conference (BMVC).*

[6] Deng, J., Guo, J., Xue, N., & Zafeiriou, S. (2019). ArcFace: Additive Angular Margin Loss for Deep Face Recognition. *IEEE Conference on Computer Vision and Pattern Recognition.*

8 Appendices

8.1 Key Classes and Functions

8.1.1 Dense Layer

The `Dense` class implements a fully connected layer with the following methods:

- `forward(A)`: Performs forward propagation: $Z = AW + b$.
- `backward(dZ)`: Computes gradients for backward propagation.
- `update(learning_rate)`: Updates parameters using gradient descent.

8.1.2 Activation Functions

The `Activation` class supports multiple activation functions:

- **Sigmoid**: $\sigma(z) = \frac{1}{1+e^{-z}}$ for binary classification output.

- **ReLU**: $\mathrm{ReLU}(z) = \max(0, z)$ for hidden layers.
- **Softmax**: $\mathrm{softmax}(z_i) = \frac{e^{z_i}}{\sum_j e^{z_j}}$ for multi-class classification.

8.1.3 Convolutional Layer

The `Conv` class implements convolutional operations with forward propagation, backward propagation, and parameter update functionality.

8.1.4 Max Pooling Layer

The `MaxPool` class implements max pooling with forward and backward propagation methods.

8.2 Hyperparameter Settings

Table 5 summarizes the hyperparameters used for training both models.

Table 5: Training Hyperparameters

Hyperparameter	Neural Network	CNN
Learning Rate	0.001	0.001
Batch Size	64	64
Epochs	15	20
Dropout Rate	0.5	0.5
Loss Function	BCEWithLogitsLoss	BCEWithLogitsLoss
Optimizer	Adam	Adam

8.3 Data Preprocessing Pipeline

The preprocessing pipeline consists of the following steps: (1) RGB color preservation to maintain texture and shadow information, (2) face cropping, (3) resizing to 128×128 pixels, (4) normalization using mean=[0.5, 0.5, 0.5] and std=[0.5, 0.5, 0.5] to scale pixel values to [-1, 1], and (5) data augmentation including horizontal flipping, rotation (±15°), brightness adjustment, and contrast adjustment. For the Neural Network model, RGB images are first flattened to 49,152-dimensional vectors ($128 \times 128 \times 3$), then reduced to 256 dimensions using PCA with whitening to create manageable feature vectors. For the CNN model, images maintain their 3D spatial structure with dimensions $(128, 128, 3)$.