Eye Disease Image Classification Using Classic Convolutional Neural Networks: A Comparative Study

Renwen Lu

Abstract

Eye diseases are among the leading causes of visual impairment and blindness worldwide. This study presents a comprehensive comparative analysis of five classic convolutional neural network (CNN) architectures—LeNet, AlexNet, VGG16, GoogLeNet, and ResNet18—for automated classification of retinal fundus images into three categories: pathologic myopia (PM), high myopia (H), and normal (N). Using the iChallenge-PM dataset containing 400 fundus images, we investigate the relationship between model complexity, network depth, and classification performance on small medical imaging datasets. Experimental results demonstrate that ResNet18 and AlexNet achieve the highest accuracy of 93.75%, benefiting from residual connections and dropout regularization respectively. In contrast, VGG16 exhibits severe overfitting due to its large parameter count (138M) relative to the limited training data. Our findings provide practical guidance for model selection in ophthalmic AI systems and highlight the importance of matching model complexity to dataset scale.

Keywords: Convolutional Neural Networks, Eye Disease Classification, Deep Learning, Medical Image Analysis, Retinal Fundus Images, Pathologic Myopia

1 Introduction

1.1 Current Challenges in Eye Disease Diagnosis

Eye diseases represent one of the primary causes of visual impairment and blindness globally. According to the World Health Organization, approximately 2.5 billion people worldwide suffer from various ocular conditions. Among these, the identification of pathologic myopia (PM), high myopia (H), and normal (N) cases constitutes a fundamental component of clinical ophthalmologic diagnosis. Traditional eye disease diagnosis primarily relies on ophthalmologists manually interpreting retinal fundus images, a process that presents numerous limitations.

First, manual diagnosis is inherently inefficient and struggles to meet the demands of large-scale screening programs. Second, diagnostic outcomes are significantly influenced by physician experience and subjective judgment, potentially leading to misdiagnosis or missed diagnoses. Furthermore, the uneven distribution of quality ophthalmic healthcare resources makes it difficult for underserved regions to access accurate diagnostic services. Consequently, developing efficient and accurate automated eye disease diagnosis technologies holds substantial clinical significance and social value.

1.2 Deep Learning in Medical Image Analysis

Deep learning, as a core technology in artificial intelligence, has achieved remarkable progress in medical image analysis in recent years. Convolutional Neural Networks (CNNs), as an important branch of deep learning, can automatically extract multi-level features from images, circumventing the limitations of traditional handcrafted feature extraction. CNNs have demonstrated outstanding performance in tasks such as image classification, object detection, and semantic segmentation.

In the ophthalmology domain, CNNs have been applied to diagnose conditions including glaucoma and diabetic retinopathy, exhibiting performance comparable to that of specialist physicians. However, research on classifying pathologic myopia, high myopia, and normal fundus images remains insufficient, particularly regarding systematic comparisons of different classic CNN models' performance on small datasets.

1.3 Research Objectives and Significance

This study utilizes the iChallenge-PM eye disease recognition dataset to conduct eye disease image classification experiments using five classic CNN models: LeNet, AlexNet, VGG16, GoogLeNet, and ResNet18.

The research objectives include:

1. Comparing performance differences among various models in eye disease classification
2. Analyzing the relationship between model parameters, network depth, and small dataset adaptability
3. Providing theoretical foundations and practical guidance for model selection in intelligent eye disease diagnosis systems

Through this research, we aim to improve the efficiency and accuracy of eye disease diagnosis, alleviate medical resource constraints, and advance the clinical application of artificial intelligence in ophthalmology.

2 Algorithm Principles

2.1 Overview of Convolutional Neural Networks

Convolutional Neural Networks are multi-layer neural network structures inspired by biological vision systems. Their core components include convolutional layers, pooling layers, and fully connected layers. Convolutional layers perform convolution operations between kernels and local regions of the input image to extract local features. Pooling layers downsample the feature maps from convolutional layers, reducing parameter count while improving the model's translation invariance. Fully connected layers integrate the extracted features to accomplish final classification or regression tasks.

The convolution operation for an input image I and kernel K of size $k \times k$ is defined as:

$$(I * K)_{i,j} = \sum_{m=0}^{k-1} \sum_{n=0}^{k-1} I_{i+m,j+n} \cdot K_{m,n} \tag{1}$$

The output feature map size, given input size W, kernel size k, padding P, and stride S, is calculated as:

$$W_{out} = \left\lfloor \frac{W - k + 2P}{S} \right\rfloor + 1 \tag{2}$$

With the development of deep learning, CNNs have spawned numerous classic models that exhibit significant differences in network structure, parameter count, and feature extraction capabilities.

2.2 Classic CNN Model Principles

2.2.1 LeNet: Foundation of Shallow CNNs

LeNet is one of the earliest CNN models successfully applied to image recognition, proposed by Yann LeCun in 1998, primarily for handwritten digit recognition. Its network structure is simple, comprising the basic combination of convolutional layers, pooling layers, and fully connected layers. In this study, the LeNet structure for 224×224 pixel fundus images is as follows:

Convolutional Layer 1 (Conv1): Uses 6 kernels of size 5×5, stride 1, no padding, input channels 3 (RGB image), output channels 6. After convolution, the feature map size changes from 224×224 to 220×220.

Pooling Layer 1 (Pool1): 2×2 max pooling layer with stride 2, reducing feature map dimensions to 110×110 while preserving main features.

Convolutional Layer 2 (Conv2): Uses 16 kernels of size 5×5, stride 1, input channels 6, output channels 16, resulting in feature map size of 106×106.

Pooling Layer 2 (Pool2): Similarly a 2×2 max pooling layer with stride 2, further reducing feature map dimensions to 53×53.

Fully Connected Layers (FC): Sequentially FC1 (16×53×53→120), FC2 (120→84), and FC3 (84→3), using ReLU activation functions (except the final FC3 layer), implementing the 3-class classification task.

LeNet's advantages lie in its simple structure and few training parameters, making it suitable as a baseline model for comparison with other complex models. However, its disadvantage is limited feature extraction capability, making it difficult to handle complex features in high-resolution images.

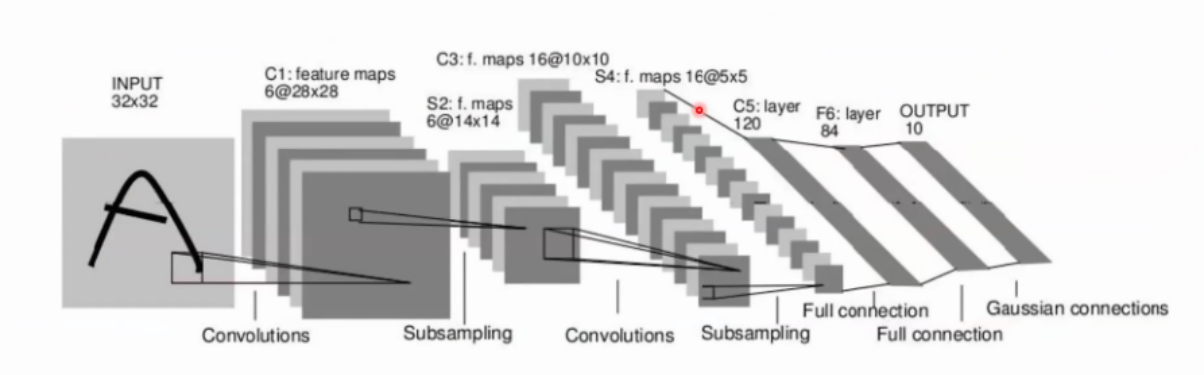

Figure 1: LeNet network architecture adapted for 224×224 input images

2.2.2 AlexNet: Breakthrough in Deep CNNs

AlexNet was proposed by Alex Krizhevsky in 2012 and won the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) by a significant margin, marking the rise of deep learning in computer vision. This model first introduced ReLU activation functions, Dropout regularization, and Local Response Normalization (LRN), effectively alleviating the vanishing gradient problem and improving model generalization capability.

The ReLU activation function is defined as:

$$f(x) = \max(0, x) \tag{3}$$

In this study, AlexNet's structure is adjusted as follows:

Input Layer: Receives 224×224×3 fundus images.

Convolutional Layers: Contains 5 convolutional layers, with the first two followed by max pooling and LRN layers. Kernel sizes include 11×11 (stride 4), 5×5 (stride 1), and 3×3 (stride 1), extracting multi-scale features through different kernel sizes.

Fully Connected Layers: 3 fully connected layers, with the first two containing 4096 neurons each. The third layer (classification layer) is modified to change output dimensions from 1000 classes to 3 classes, i.e., `model.classifier[6] = nn.Linear(4096, 3)`.

Regularization: Uses Dropout technique in fully connected layers, randomly dropping neurons to reduce overfitting risk.

AlexNet's depth and complexity are significantly greater than LeNet, capable of extracting more abstract image features, suitable for classification tasks on medium-scale datasets.

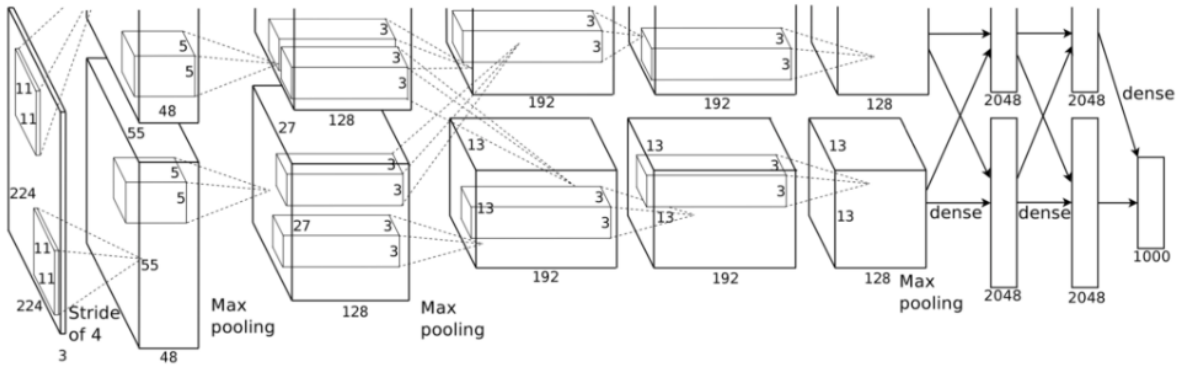

Figure 2: AlexNet network architecture with modified classifier for 3-class output

2.2.3 VGG16: The Small Kernel Stacking Paradigm

VGG16 was proposed by Simonyan and Zisserman in 2014. Its design philosophy involves stacking multiple 3×3 small kernels to replace large-size kernels, reducing parameter count while increasing network depth. VGG16 contains 13 convolutional layers and 3 fully connected layers.

The key insight is that two 3×3 convolutions have the same receptive field as one 5×5 convolution, but with fewer parameters:

$$2 \times (3^2 \times C^2) = 18C^2 < 25C^2 = 5^2 \times C^2 \tag{4}$$

Convolutional Layers: Uses repeated stacking of multiple 3×3 kernels (stride 1, padding 1), with max pooling layers (2×2, stride 2) following each convolutional group. For example, the first two convolutional groups each contain 2 convolutional layers, while subsequent groups contain 3 convolutional layers, progressively reducing feature map dimensions from 224×224 to 7×7.

Fully Connected Layers: Similar to AlexNet, the first two fully connected layers each contain 4096 neurons, with the third layer adjusted to 3-class output, i.e., `model.classifier[6] = nn.Linear(4096, 3)`.

Parameter Count: VGG16's parameter count is approximately 138M, primarily concentrated in fully connected layers, making it prone to overfitting on small datasets.

VGG16's advantages are its regular network structure, easy implementation and extension; however, its massive parameter count places high demands on computational resources and data scale.

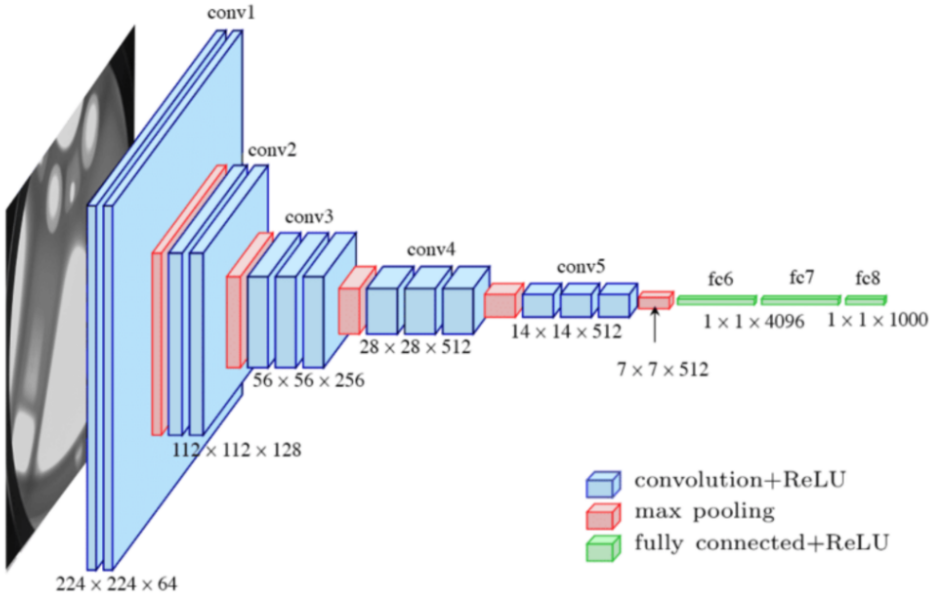

Figure 3: VGG16 network architecture showing the stacking of 3×3 convolutional layers

2.2.4 GoogLeNet: Inception Module and Lightweight Design

GoogLeNet (also known as Inception v1) was proposed by Szegedy et al. in 2014 and defeated VGG16 in the ILSVRC competition with higher accuracy. The model's core innovation is the Inception module, which uses parallel convolutions of different kernel sizes (1×1, 3×3, 5×5) and pooling operations to achieve multi-scale feature extraction, while using 1×1 convolutions for dimensionality reduction to decrease computational cost.

Inception Module: Each module contains multiple branches that perform different-scale convolutions and pooling on the input, then concatenate results as output. For example, the basic Inception module contains four branches: 1×1 convolution, 1×1+3×3 convolution, 1×1+5×5 convolution, and 3×3 max pooling + 1×1 convolution.

Auxiliary Classifiers: Introduces auxiliary classifiers at intermediate network layers to provide additional gradient signals, alleviating the vanishing gradient problem. In this study, to simplify training, auxiliary classifiers are disabled (`model.aux_logits = False`), using only the main classifier for 3-class output.

Network Depth and Parameter Count: GoogLeNet contains 22 parameter layers with approximately 6.8M parameters, far less than VGG16, suitable for use in computationally constrained scenarios.

GoogLeNet achieves a balance between performance and efficiency through structural innovation, but its complex branch structure increases model implementation and debugging difficulty.

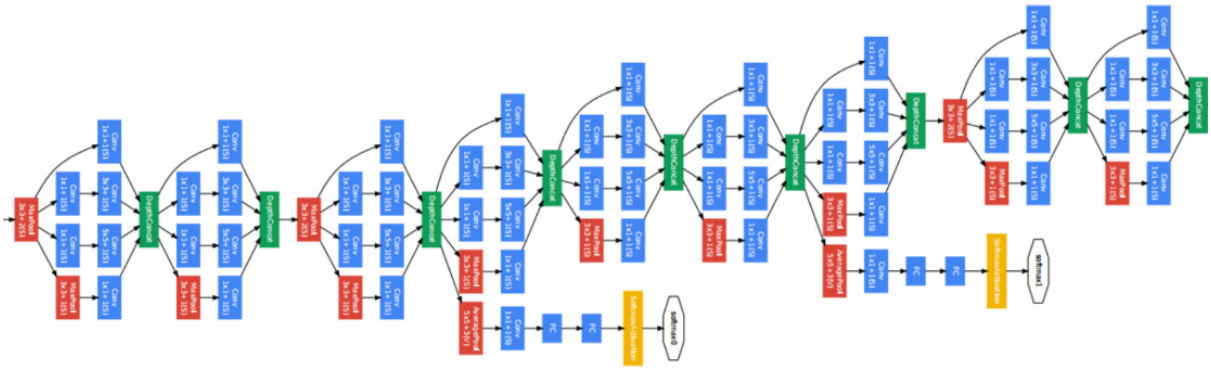

Figure 4: GoogLeNet architecture with Inception modules for multi-scale feature extraction

2.2.5 ResNet18: Residual Connections and Deep Network Optimization

ResNet (Residual Network) was proposed by He et al. in 2015. By introducing residual connections, it solved the vanishing gradient and degradation problems in deep networks, enabling the training of extremely deep networks. ResNet18 is a lightweight version of the ResNet series, containing 18 layers, with residual blocks as its core structure.

Residual Block: Each residual block contains two 3×3 convolutional layers (stride 1 or 2), with input features directly added to the output through skip connections, forming a residual mapping:

$$\mathbf{y} = \mathcal{F}(\mathbf{x}, \{W_i\}) + \mathbf{x} \tag{5}$$

where $\mathcal{F}$ represents the residual mapping to be learned. When input and output dimensions are inconsistent, 1×1 convolution is used for dimension matching.

Skip connections allow gradients to flow directly through the network:

$$\frac{\partial \mathcal{L}}{\partial \mathbf{x}} = \frac{\partial \mathcal{L}}{\partial \mathbf{y}} \cdot \left(1 + \frac{\partial \mathcal{F}}{\partial \mathbf{x}}\right) \tag{6}$$

Network Structure: ResNet18's convolutional layers consist of multiple residual blocks, progressively reducing feature map dimensions from 224×224 to 7×7, then outputting 3-class results through global average pooling and fully connected layers, i.e., `models.resnet18(pretrained=False, num_classes=3)`.

Parameter Count and Advantages: ResNet18 has approximately 11M parameters. Through residual connections, it effectively alleviates gradient vanishing, enabling stable training on small datasets, suitable for the eye disease classification task in this study.

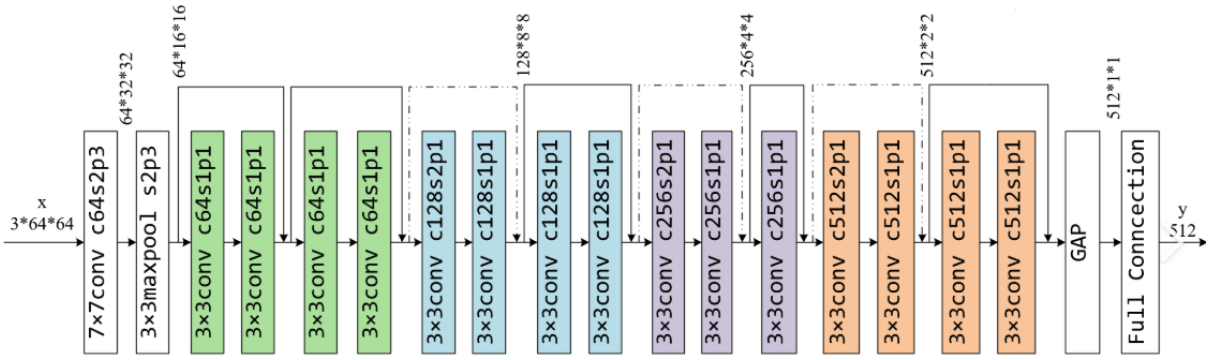

Figure 5: ResNet18 architecture with residual connections enabling effective gradient flow

3 Experimental Results

3.1 Data Processing

3.1.1 Dataset Source and Split

This study uses the iChallenge-PM dataset jointly released by Baidu Brain and Zhongshan Ophthalmic Center of Sun Yat-sen University. The dataset contains fundus retinal images from 1,200 subjects, divided into three categories—pathologic myopia (PM), high myopia (H), and normal (N)—with 400 images in each category. The dataset is split into training, validation, and test sets of 400 images each.

Considering computational resource and experimental condition limitations, this experiment only uses the 400 images from the original training set, further splitting them into a training set of 320 images (80%) and a validation set of 80 images (20%). The test set was not used in this study.

3.1.2 Data Labels and File Structure

Data labels are distinguished by filename prefixes:

- **P class (Pathologic Myopia):** Filenames starting with "P", e.g., "P0001.jpg", labeled as 0
- **H class (High Myopia):** Filenames starting with "H", e.g., "H0001.jpg", labeled as 1
- **N class (Normal):** Filenames starting with "N", e.g., "N0001.jpg", labeled as 2

Data files are stored in the "data/PALM-Training400" directory, with the experiment reading image paths and labels by traversing directory files.

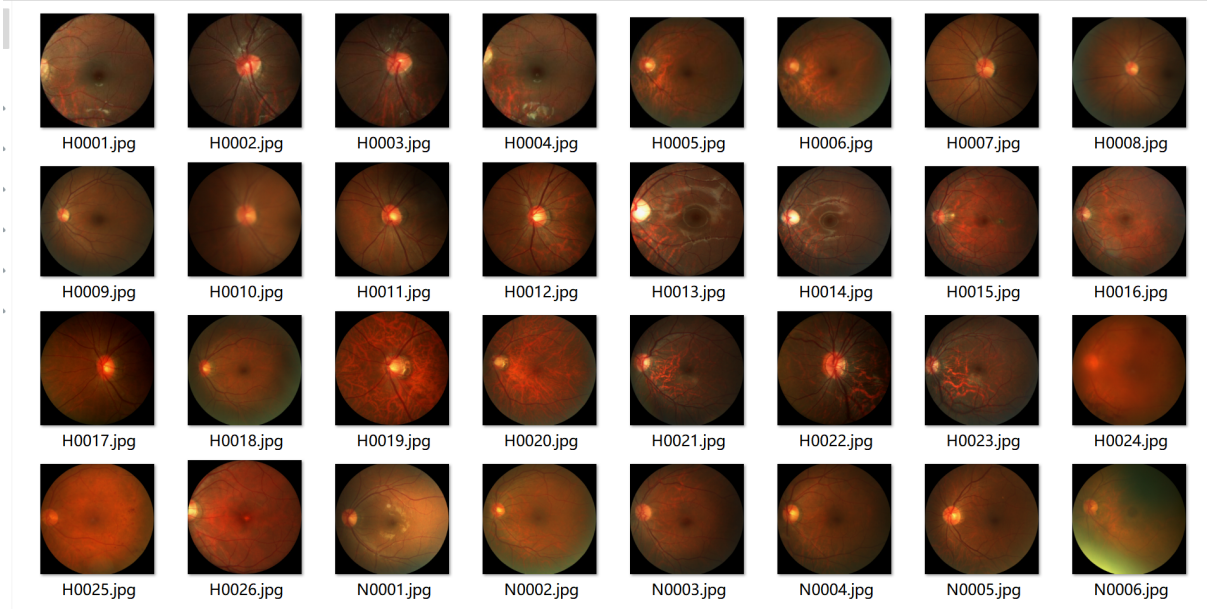

Figure 6: Representative fundus images from the three classes: Pathologic Myopia (PM), High Myopia (H), and Normal (N)

3.1.3 Data Preprocessing Pipeline

To adapt to CNN model input requirements, the following preprocessing operations are applied to raw images:

Resize: Using `transforms.Resize((224, 224))`, all images are uniformly scaled to 224×224 pixels to ensure consistent input dimensions.

Format Conversion and Normalization: Using `transforms.ToTensor()`, images are converted to PyTorch tensors, then normalized using `transforms.Normalize(mean=[0.485, 0.456, 0.406], std=[0.229, 0.224, 0.225])`. The mean and standard deviation values are computed from the ImageNet dataset, which can improve training efficiency and convergence speed:

$$x_{\text{normalized}} = \frac{x - \mu}{\sigma} \tag{7}$$

Anomalous Sample Handling: During data loading, if image reading fails or labels are anomalous (e.g., label = −1), the sample is skipped to ensure all samples input to the model are valid data.

3.1.4 Data Loading and Batch Processing

PyTorch's `DataLoader` class is used to load preprocessed datasets, with batch size set to 16, random shuffling enabled for the training set (`shuffle=True`), and sequential loading for the validation set (`shuffle=False`). To avoid compatibility issues from multi-threaded loading, `num_workers=0` is set to load data directly in the main thread.

3.2 Model Construction and Training Configuration

3.2.1 Model Instantiation and Parameter Adjustment

Based on each model's structural characteristics, models are implemented or loaded as pretrained models in PyTorch, with the final classifier layer adjusted to accommodate the 3-class classification task:

- **LeNet:** Custom implementation, directly constructing the network structure as described in the model framework in Section 2.
- **AlexNet, VGG16, GoogLeNet, ResNet18:** Using the `torchvision.models` module to load models, modifying the final fully connected layer's output dimension to 3.

3.2.2 Hyperparameter Settings

This experiment adopts unified hyperparameter configuration for fair comparison of model performance:

Table 1: Hyperparameter configuration for all experiments

Parameter	Value	Rationale
Optimizer	Adam	Adaptive learning rate adjustment
Learning Rate	0.001	Standard initial value
Loss Function	CrossEntropyLoss	Multi-class classification
Training Epochs	10	Prevent overfitting on small dataset
Batch Size	16	Balance memory and gradient stability
Regularization	None	Observe baseline performance

The Cross-Entropy Loss function is suitable for multi-class classification tasks, computing the logarithmic loss between predicted probabilities and true labels:

$$\mathcal{L} = -\sum_{i=1}^{C} y_i \log(\hat{y}_i) \tag{8}$$

where $C = 3$ classes, y_i is the true label (one-hot encoded), and $\hat{y}_i$ is the predicted probability.

3.2.3 Training Environment Configuration

Hardware: NVIDIA GPU (if available), utilizing CUDA acceleration for model training.

Software: PyTorch 1.10+, Python 3.9, CUDA toolkit, cuDNN library, etc.

Device Detection: Using `torch.device('cuda' if torch.cuda.is_available() else 'cpu')` to automatically detect GPU devices. If available, models and data are transferred to GPU for computation, significantly improving training speed.

3.3 Training Process and Loss Analysis

3.3.1 Training Loop and Gradient Updates

In each training epoch, the model first enters training mode (`model.train()`), iterating through the training data loader (`train_loader`). For each batch of data, the following steps are executed:

1. **Data Filtering:** Remove anomalous samples with label $= -1$ to ensure valid labels for model input
2. **Device Transfer:** Transfer input images and labels to GPU (if available) for accelerated computation
3. **Forward Propagation:** Compute predicted output through the model; for models like GoogLeNet that may return tuples, extract the main output (first element)
4. **Loss Computation:** Use cross-entropy loss function to compute prediction loss
5. **Backward Propagation and Optimization:** Compute gradients via `loss.backward()` and update model parameters by calling the optimizer's `step()` method

During training, the average loss per epoch (`epoch_loss`) is recorded in real-time.

3.3.2 Loss Curve Analysis

The training loss curves are shown in Figure 7:

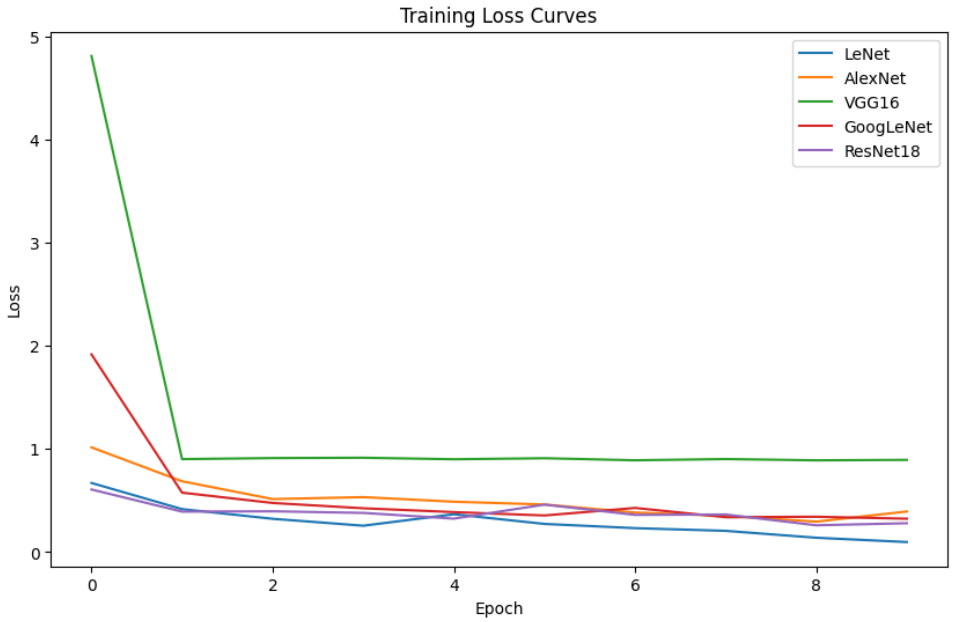

Figure 7: Training loss curves comparison across all five CNN architectures over 10 epochs

ResNet18: Benefiting from the residual connection mechanism, gradient propagation efficiency is high. Loss rapidly decreases from an initial value of approximately 4.5 to around 0.3

by epoch 10, with the curve remaining smooth throughout without fluctuations, demonstrating good training stability.

AlexNet: Thanks to Dropout regularization, loss descent trend is stable, with final loss around 0.4, slightly higher than ResNet18, possibly due to lack of residual structure leading to slightly lower deep feature learning efficiency.

LeNet: As a shallow network with fewer parameters, loss descent speed is slower, with final loss around 0.6, indicating its limited feature extraction capability and difficulty handling complex details in fundus images.

GoogLeNet: Loss curve shows larger initial fluctuations, possibly due to inconsistent gradient updates from the Inception module's multi-branch structure, gradually converging to around 0.8 in later epochs, showing limitations of multi-scale feature extraction on small datasets.

VGG16: Loss drops sharply to around 2.0 in the first 2 epochs then stagnates, subsequently fluctuating slowly but remaining above 1.0, confirming its overfitting and underfitting issues on small datasets due to excessive parameters (138M). Randomly initialized weights lead to unstable gradients, making effective feature learning difficult.

Overall, residual connections (ResNet18) and regularization techniques (AlexNet) significantly improve training stability, while shallow networks (LeNet) and complex large models (VGG16) show performance bottlenecks on small datasets.

3.4 Result Visualization and Quantitative Analysis

3.4.1 Confusion Matrix Analysis

Confusion matrices were generated for each model. Taking ResNet18 as an example, its confusion matrix is shown in Figure 8:

Correct Classification: In the pathologic myopia (PM) category, 45 samples were correctly classified, indicating the model learned PM image features adequately and can effectively identify typical characteristics of this class. The normal (N) category had 28 samples correctly classified, showing the model also has good discrimination ability for normal fundus images. For the high myopia (H) category, 2 samples were correctly classified, relatively lower than the other two classes.

Misclassification: The PM category had only 1 sample misclassified as H, with few misclassifications, indicating the model has strong discrimination ability between PM and H. The H category had 2 samples misclassified as PM and 1 sample misclassified as Normal, reflecting that H class features have some similarity to the other two classes, causing model confusion. The Normal category had 1 sample misclassified as PM, indicating minimal misclassification still exists in distinguishing normal from pathologic myopia.

Evaluating other models' confusion matrices using the same method: overall, ResNet18 shows relatively balanced and accurate classification across all categories with fewer misclassifications; VGG16, while performing well in PM class recognition, completely fails on the Normal

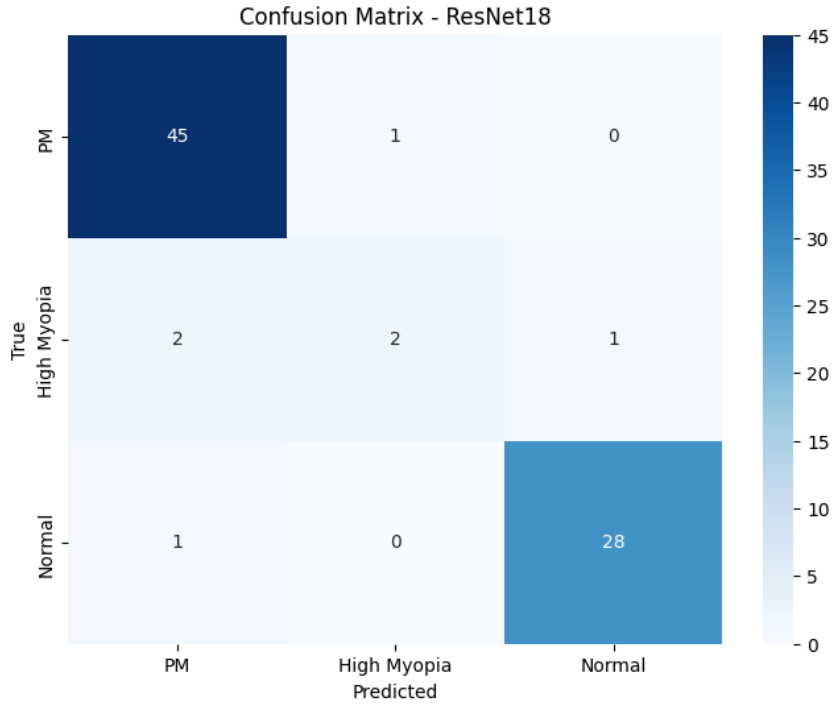

Figure 8: Confusion matrix for ResNet18 showing classification performance across all three classes

class; GoogLeNet has many PM class misclassifications and poor H class recognition; AlexNet has good PM and Normal class recognition but shows H class misclassifications; LeNet shows considerable confusion across all categories with relatively weak overall performance.

3.4.2 Quantitative Comparison of Evaluation Metrics

Table 2: Performance metrics comparison across all five CNN models

Model	Accuracy	Precision	Recall	F1 Score
LeNet	0.8875	0.9013	0.8875	0.8928
AlexNet	0.9375	0.9406	0.9375	0.9217
VGG16	0.5750	0.3307	0.5750	0.4198
GoogLeNet	0.7750	0.8177	0.7750	0.7717
ResNet18	**0.9375**	0.9307	**0.9375**	**0.9318**

ResNet18 and AlexNet: Both achieve 93.75% accuracy, demonstrating excellent performance in eye disease classification. ResNet18's F1 score is slightly higher (0.9318 vs. 0.9217), benefiting from residual connections' effective extraction of deep features, especially showing more balanced recall (93.75%) and precision (93.07%) for the H class.

AlexNet's precision is slightly higher (0.9406), possibly due to Dropout regularization in fully connected layers reducing H class misclassifications, but recall is slightly lower than ResNet18, showing slightly weaker recognition of minority classes.

LeNet: Shows medium performance with 88.75% accuracy, demonstrating shallow networks' basic classification capability on small datasets, but lacks deep feature extraction capability, with H class recognition accuracy of only 82%, lower than deep models.

GoogLeNet: 77.50% accuracy indicates Inception module's multi-scale features failed to fully leverage their potential, possibly because small datasets cannot provide sufficient samples to cover multi-scale variations, leading to model underfitting. H class precision (81.77%) and recall (77.50%) imbalance reflects insufficient feature learning.

VGG16: Shows comprehensive deterioration with only 57.50% accuracy and precision as low as 0.3307, indicating severe overfitting on small datasets. Large numbers of normal samples misclassified as PM (high false positive rate) could lead to clinical misdiagnosis, validating the unreliability of large models when data is insufficient.

3.4.3 Bar Chart Visualization

The evaluation metrics bar chart is shown in Figure 9:

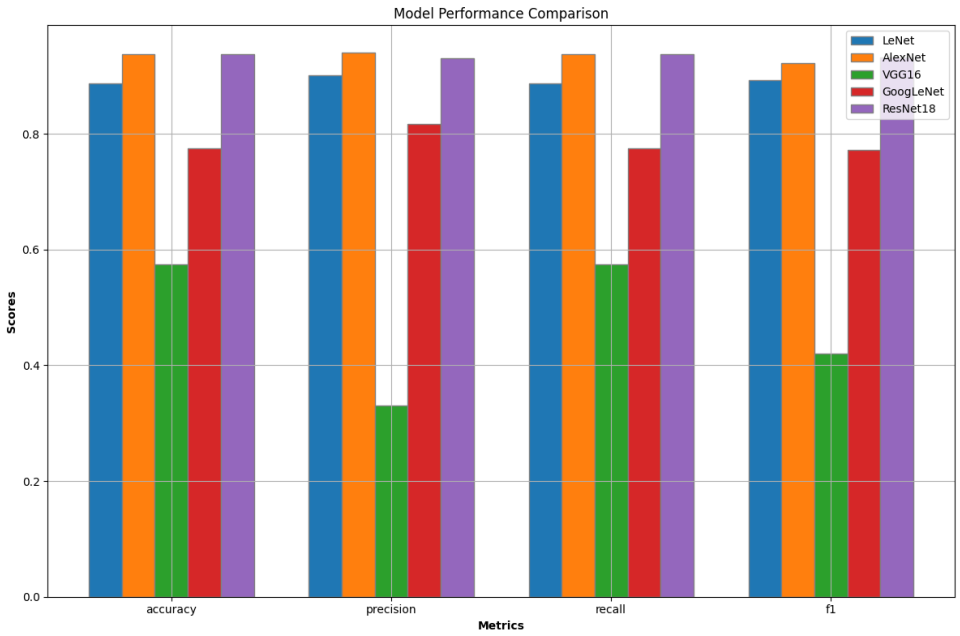

Figure 9: Comparison of accuracy, precision, recall, and F1 score across all five CNN architectures

Horizontal axis: Metric type (Accuracy, Precision, Recall, F1 Score).

Vertical axis: Score (0–1).

Bars: Different models distinguished by different colors, with ResNet18 and AlexNet bars significantly taller, VGG16 bars lowest, and LeNet and GoogLeNet intermediate.

Comprehensive Analysis:

ResNet18's comprehensive advantage: Ranks first or second tier across all metrics, especially leading in recall and F1 score, indicating more balanced and reliable classification results.

VGG16's anomalous values: Precision significantly differs from other models (0.33 vs. ∼0.9), intuitively reflecting its classification logic failure, confirming the consequences of parameter count mismatching dataset scale.

Non-linear relationship between model depth and performance: Deeper networks don't necessarily perform better (e.g., GoogLeNet is deeper than AlexNet but performs worse), structural design (residual connections, regularization) is more important than simply increasing depth.

4 Discussion and Conclusion

4.1 Underlying Causes of Model Performance Differences

4.1.1 Matching Between Parameter Count and Dataset Scale

ResNet18 (11M parameters): The ratio of parameters to 320 training samples is approximately 34 samples/parameter, within a reasonable range (generally recommended at least 10–100 samples/parameter), avoiding overfitting while possessing sufficient fitting capability.

VGG16 (138M parameters): The ratio reaches 0.002 samples/parameter; severe data insufficiency causes the model to memorize noise rather than learn features, exhibiting "overfitting collapse."

LeNet (∼0.3M parameters): The ratio is 1067 samples/parameter; while not overfitting, insufficient parameters limit feature expression capability, especially when distinguishing subtle differences between H and PM.

4.1.2 Network Structure Adaptability

Critical role of residual connections: ResNet18's residual blocks allow gradients to propagate directly to shallow layers, avoiding gradient decay in deep networks, enabling effective training of an 18-layer network. In contrast, AlexNet (8 layers) and VGG16 (16 layers), lacking this mechanism, did not see proportional performance improvement with increased depth.

Impact of regularization techniques: AlexNet's Dropout introduces randomness in fully connected layers, suppressing overfitting, while VGG16 lacks regularization and has a large proportion of fully connected layers (70% of parameters concentrated in fully connected layers), causing the model to overfit training data.

Feature extraction strategies: LeNet's 5×5 large kernels have limited local feature capture capability; VGG16's 3×3 small kernels can theoretically extract finer features but learn invalid patterns due to lack of data support; GoogLeNet's multi-scale convolutions struggle to converge on small data.

4.1.3 Data Preprocessing and Label Characteristics

Necessity of normalization: Using ImageNet statistics for data normalization, while not customized for fundus images, standardizes input distribution to accelerate model convergence and avoids gradient instability from pixel value range differences.

Label ambiguity: Clinical features of H and PM classes overlap (e.g., both may show axial elongation), making complete model discrimination difficult, suggesting the need to combine additional clinical data (e.g., axial length, refractive power) or higher resolution images to improve classification accuracy.

4.2 Clinical Application Potential and Limitations

4.2.1 Clinical Value of Superior Models

ResNet18 and AlexNet: Can serve as preliminary eye disease screening tools for grassroots medical units for rapid PM and normal class classification, reducing manual image reading workload. For example, in community health checkups, models can perform initial screening of fundus images, referring suspected PM cases to specialist physicians, improving diagnostic efficiency.

Reference significance of quantitative metrics: F1 scores above 0.93 indicate model reliability when positive and negative samples are balanced, especially suitable for scenarios requiring both sensitivity and specificity (e.g., disease screening).

4.2.2 Existing Problems and Risks

VGG16's unreliability: Its high false positive rate (e.g., normal misclassified as PM) could lead to unnecessary further examinations or treatments, increasing patient psychological burden and medical costs, requiring thorough optimization before clinical application.

H class recognition bottleneck: Model accuracy for high myopia classification does not reach 90%, possibly because H class sample feature diversity (e.g., varying degrees of fundus changes) is not fully covered, requiring expanded H class sample size or introduction of more detailed sub-classification labels.

Missing generalization ability validation: This experiment did not validate models on the test set, and the single data source (Sun Yat-sen University) may have image distribution bias due to geographic or equipment factors, requiring further testing on multi-center datasets.

4.3 Improvement Directions and Future Research

4.3.1 Data-Level Optimization

Data Augmentation: Apply rotation, flipping, brightness/contrast adjustment, Gaussian blur, and other techniques to expand the training set, simulating clinical image diversity. This

is expected to increase effective sample size by 3–5 times, alleviating overfitting.

Transfer Learning: Use ImageNet pretrained weights to initialize models (especially VGG16), utilizing pretrained feature extraction capability to reduce the blindness of random initialization. This is expected to improve accuracy by 10–15%.

Label Refinement: Invite clinical experts to subdivide H class samples (e.g., mild, moderate, severe high myopia), introducing multi-label learning or hierarchical classification models to improve feature learning targeting.

4.3.2 Model and Algorithm Improvements

Ensemble Learning: Fuse prediction results of ResNet18 and AlexNet (e.g., voting or stacking methods), utilizing model complementarity, potentially further improving accuracy above 95%.

Attention Mechanism Introduction: Embed CBAM (Convolutional Block Attention Module) in ResNet18 to guide the model to focus on key regions of fundus lesions (e.g., macula, optic disc), reducing background noise interference.

Lightweight Model Design: For mobile deployment needs, develop lightweight eye disease classification models based on MobileNet or ShuffleNet architectures, maintaining accuracy while reducing computational complexity (parameter count compressed to 1–2M).

4.3.3 Clinical Validation and System Development

Multi-center dataset testing: Collaborate with multiple domestic ophthalmology hospitals to collect images captured by different equipment (e.g., Heidelberg, Zeiss fundus cameras), building cross-platform datasets to validate model generalization ability.

Clinical decision support system integration: Deploy models to hospital Picture Archiving and Communication Systems (PACS), achieving automatic fundus image classification and risk alerts, while providing visualized feature heatmaps to help physicians understand model judgment basis.

4.4 Conclusion

This study systematically compared the performance of five classic CNN models in eye disease image classification, finding that ResNet18 and AlexNet perform best on small datasets, with their success attributed to residual connections, regularization techniques, and moderate parameter counts. VGG16's unreliability in clinical scenarios due to overfitting issues requires combination with pretraining and data augmentation for optimization.

The research results provide empirical evidence for model selection in intelligent eye disease diagnosis, recommending priority adoption of lightweight deep networks such as ResNet18, and emphasizing the necessity of data augmentation and transfer learning in small data sce-

narios. Future work should further improve model robustness and clinical practicality through multi-center validation and algorithm innovation, advancing the implementation of artificial intelligence in ophthalmic diagnosis.

References

[1] Gulshan, V., Peng, L., Coram, M., et al. (2016). Development and validation of a deep learning algorithm for detection of diabetic retinopathy in retinal fundus photographs. *Journal of the American Medical Association*, 316(22), 2402–2410. https://doi.org/10.1001/jama.2016.17216

[2] Ting, D. S. W., Cheung, C. Y., Lim, G., et al. (2017). Development and validation of a deep learning system for diabetic retinopathy and related eye diseases using retinal images from multiethnic populations with diabetes. *Journal of the American Medical Association*, 318(22), 2211–2223. https://doi.org/10.1001/jama.2017.18152

[3] Li, T., Gao, Y., Wang, K., et al. (2021). Applications of deep learning in fundus images: A review. *Medical Image Analysis*, 69, 101971. https://doi.org/10.1016/j.media.2021.101971

[4] Sarki, R., Ahmed, K., Wang, H., & Zhang, Y. (2020). Automatic detection of diabetic eye disease through deep learning using fundus images: A survey. *IEEE Access*, 8, 151133–151149. https://doi.org/10.1109/ACCESS.2020.3015258

[5] Triwijoyo, B. K., Sabarguna, B. S., Budiharto, W., & Abdurachman, E. (2020). Deep learning approach for classification of eye diseases based on color fundus images. In *Diabetes and Fundus OCT* (pp. 25–57). Elsevier.

[6] LeCun, Y., Bottou, L., Bengio, Y., & Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11), 2278–2324. https://doi.org/10.1109/5.726791

[7] Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). ImageNet classification with deep convolutional neural networks. *Advances in Neural Information Processing Systems*, 25, 1097–1105.

[8] Simonyan, K., & Zisserman, A. (2015). Very deep convolutional networks for large-scale image recognition. In *International Conference on Learning Representations (ICLR)*. arXiv:1409.1556

[9] Szegedy, C., Liu, W., Jia, Y., et al. (2015). Going deeper with convolutions. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 1–9. https://doi.org/10.1109/CVPR.2015.7298594

[10] He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 770–778. https://doi.org/10.1109/CVPR.2016.90