HYBRID DEEP LEARNING AND MACHINE LEARNING MODELS FOR FUNDUS LESION IMAGE CLASSIFICATION

A. Jinda Dong[⋆] *B. Zhiling Li*[⋆] *C. Zhuoyang Wang*[†]

[⋆] School of System Design and Intelligent Manufacturing
Southern University of Science and Technology
Shenzhen, China
[†] Department of Electronics Engineering
Southern University of Science and Technology
Shenzhen, China

ABSTRACT

This project explores combining deep learning and traditional machine learning for image classification, focusing on fundus lesions. We used a pre-trained ResNet18 to extract features, then applied Linear Regression, MLP, KNN, and SVM for classification, achieving promising results. Additionally, we trained ResNet18 and a CNN for feature extraction, followed by MLP for multi-class classification, which also yielded satisfactory outcomes. Our findings demonstrate that this hybrid approach enhances image classification accuracy, particularly for fundus lesions.

Index Terms— deep learning, fundus lesions, image classification, ResNet18, hybrid models

1. INTRODUCTION

Image classification is a critical task in computer vision, with numerous applications in medical diagnostics, particularly in detecting and classifying fundus lesions. Traditional machine learning techniques have been widely used for this purpose, yet they often struggle with the high-dimensional nature of image data. In recent years, deep learning models, such as Convolutional Neural Networks (CNNs), have shown remarkable performance in image analysis tasks due to their ability to learn hierarchical features from raw data automatically.

In our work, We leveraged a pre-trained ResNet18 model to extract image features, which were then classified using various traditional machine learning models, including Linear Regression Classification, Multi-Layer Perceptron (MLP), K-Nearest Neighbors (KNN), and Support Vector Machine (SVM). This hybrid approach aims to capitalize on the strengths of both deep learning and traditional techniques.

Furthermore, we trained the ResNet18 network specifically for the classification of fundus lesions utilizing fundus images. To further substantiate our findings, we also developed Convolutional Neural Networks (CNNs) to autonomously extract image features, which were subsequently classified using a Multi-Layer Perceptron (MLP) for multi-class categorization. Our experimental results demonstrated robust classification performance on the fundus images, indicating a promising and reliable approach for medical image analysis.

The following sections of this paper will detail our methodology, results and analysis of the required tasks.

Our experiments were conducted on a system equipped with an AMD Ryzen R7 6800H processor and an NVIDIA RTX 3050 Ti graphics card, running on the Windows 11 Home operating system.

2. FORMULAS

Convolution operations in CNNs:

$$f(x,y) * g(x,y) = \sum_{i=-a}^{a}\sum_{j=-b}^{b} f(i,j)\cdot g(x-i, y-j) \quad (1)$$

Max pooling operation:

$$P_{x,y} = \max_{i,j\in \text{Region}} (I_{x+i,y+j}) \quad (2)$$

Cross-entropy loss for multi-class classification:

$$L = -\sum_{c=1}^{M} y_{o,c}\log(p_{o,c}) \quad (3)$$

Stochastic Gradient Descent (SGD) update rule:

$$\theta = \theta - \eta\cdot\nabla_\theta J(\theta; x^{(i)}, y^{(i)}) \quad (4)$$

Accuracy calculation:

$$\text{Accuracy} = \frac{\text{Number of Correct Predictions}}{\text{Total Number of Predictions}} \quad (5)$$

True Positive Rate and False Positive Rate for ROC curve:

$$\text{TPR} = \frac{TP}{TP+FN},\ \text{FPR} = \frac{FP}{FP+TN} \quad (6)$$

3. RELATED WORK

3.1. Residual Neural Networks

Residual Neural Networks (ResNets) represent a significant breakthrough in the training of very deep neural networks. Introduced by He et al. in their seminal 2015 paper, ResNets addresses the problem of vanishing gradients—a common issue in traditional deep networks as depth increases—by introducing residual blocks with skip connections. These connections allow gradients to flow directly through the network, effectively alleviating the problem of gradient disappearance during backpropagation [1].

The core idea behind a ResNet is to learn residual functions with reference to the layer inputs, as opposed to learning unreferenced functions. Mathematically, this can be expressed as follows:

$$\mathbf{y} = \mathcal{F}(\mathbf{x}, \{W_i\}) + \mathbf{x} \tag{7}$$

Here, $\mathbf{x}$ and $\mathbf{y}$ are the input and output of a residual block, respectively, and $\mathcal{F}(\mathbf{x}, \{W_i\})$ represents the residual mapping to be learned. For each residual block, the function $\mathcal{F}$ represents the stacked non-linear transformations, and $\{W_i\}$ denotes the set of weights associated with these transformations. The operation $\mathbf{x} + \mathcal{F}(\mathbf{x}, \{W_i\})$ is performed by a shortcut connection and element-wise addition [2].

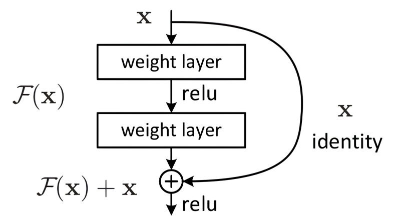

Fig. 1. The architecture of a Residual Network (ResNet) block is shown, where the input `x` is processed through two paths. In one, `x` bypasses two weight layers and is added to their output before passing through a ReLU activation. This design supports deep network training by leveraging shortcut connections for identity mapping.

ResNets have shown remarkable performance on various challenging tasks such as image classification, object detection, and segmentation, significantly reducing error rates compared to previous architectures. This model has been extensively used and adapted in numerous applications, demonstrating its versatility and efficiency in handling problems associated with deep learning architectures [1].

3.2. Convolutinoal Neuron Networks

Convolutional Neural Networks (CNNs) are a class of deep learning models designed specifically for image data processing. Their structure typically includes multiple convolutional and pooling layers that extract local features from images through convolution operations and reduce the spatial dimensions of feature maps through pooling operations, thus decreasing computational complexity. Finally, the network performs classification through a series of fully connected layers, outputting the probability distribution for each class. CNNs excel in tasks such as image classification, object detection, and image segmentation, effectively capturing hierarchical features of images.[3, 4]

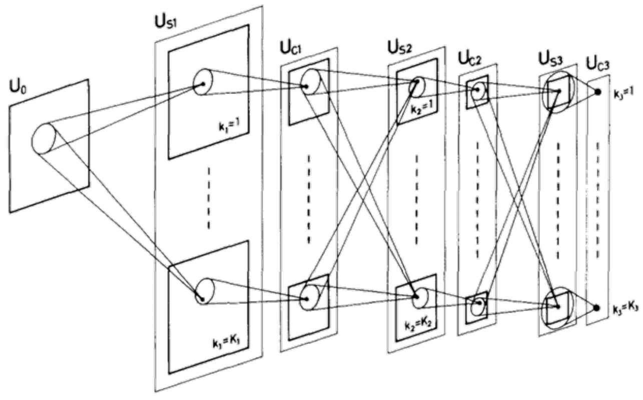

Fig. 2. Structure for the CNN model. This graph illustrates the original structure of a CNN model in 1980, where the `Us` layers represent the convolution layers, and the `Uc` layers stands for the pooling layers.[3]

4. TASK1: SIMPLE COMBINATION OF DEEP LEARNING AND TRADITIONAL MACHINE LEARNING

4.1. Methodology

In this task, we utilized a pre-trained ResNet18 model to extract 1000 features from the input images. These features were then used as inputs for classification using Linear Classification (LE), K-Nearest Neighbors (KNN), and Support Vector Machine (SVM) models, implemented with the `sklearn` library. The models were trained on a dataset of fundus images categorized into three classes.

4.2. Results

We evaluated the accuracy of each model in classifying the images and generated ROC curves to assess their performance. As shown in Fig. 3 and Table 1, the models demonstrated high classification accuracy. The area under the ROC curves approached 1, indicating excellent classification performance.

Model	LE	KNN	SVM
Accuracy	100%	100%	100%

Table 1. Classification accuracy of different models

4.3. Result Analysis

Why the results are that good? In fact, the features we use are extracted by the ResNet18, which had been examined to perform well in classification. The boundaries of these features are so clear that our classifier can easily distinguish them.

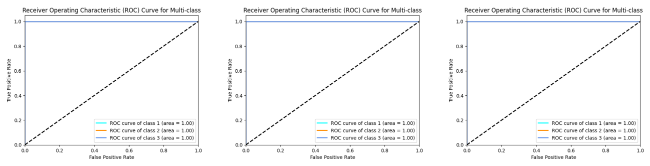

Fig. 3. ROC curves of different models (from left to right): LE, KNN, and SVM. The area under each curve is 1, indicating perfect classification performance by all three models.

5. TASK2: TRAINING THE RESNET18

5.1. Methodology

In this task, we utilized a pre-trained ResNet18 model to classify fundus images. Firstly, we defined the image preprocessing transformations, including resizing, tensor conversion, and normalization. We then loaded the training and testing datasets the `ImageFolderWithPaths` class, specifying the data paths and batch sizes. The learning rate is 0.001 and the momentum is 0.9. The size of the output layer of the ResNet is so large, so we **replace it** with a 3-class classifier.

Subsequently, we employed the default weights of the ResNet18 model and adjusted its output layer to fit our classification task. We selected the **cross-entropy** loss function and stochastic gradient descent (SGD) optimizer for model training.

We conducted multiple training iterations and observed that, with our chosen parameters, the model tends to converge after approximately three epochs, after which the loss function exhibits fluctuations. Consequently, we set the training to run for five epochs. During each epoch, the model underwent forward and backward propagation on the training set to update the weights. At the conclusion of each epoch, we evaluated the model's performance on the testing set by calculating the validation accuracy.

After training, we collected the predictions and labels from the test set. These results were used to calculate classification accuracy and plot ROC curves, providing an assessment of the model's performance.

5.2. Results

The result is pretty good. On the testing set, the accuracy of the classification is 100%, and the variation of the loss function and the final ROC curve is shown in Fig. 4.

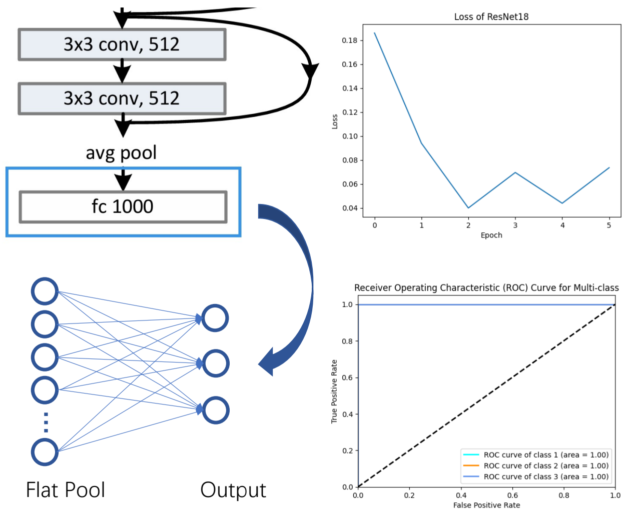

Fig. 4. Results for training and testing the ResNet18 model. The **left side** shows the replacement of the last layer of the ResNet18. We use the fully connected networks to connect the last pooling layer to the 3-class output layer, rather than `fc 1000` in that graph. On the **right side**, the upper plot shows the loss curve of the ResNet18 model during training, where the loss decreases significantly in the initial epochs and stabilizes after the third epoch, despite minor fluctuations. The lower right plot displays the ROC curves for a multi-class classification task (classifying three different types of fundus lesions), with each class achieving a perfect area under the curve (AUC) of 1.00, indicating exceptional classification performance across all categories.

5.3. Analysis

The primary reason for achieving fast convergence 100% accuracy is the superiority of the ResNet18 neural network. ResNet18 incorporates residual connections, effectively addressing the issues of vanishing and exploding gradients in deep neural networks, allowing the model to maintain high training efficiency and classification accuracy even in deeper network structures. Additionally, ResNet18 has excellent feature extraction capabilities, enabling it to extract high-quality features from complex image data, which significantly enhances classification performance. Furthermore, our model was well-trained on a specific dataset, contributing to its exceptional performance on the test set.

Model	LE	KNN	MLP	ResNet
Accuracy	96.7%	91.5%	97.6%	100%
Training Time (s)	19.2	0.362	52.5	348

Table 2. Classification accuracy and training time of different models.

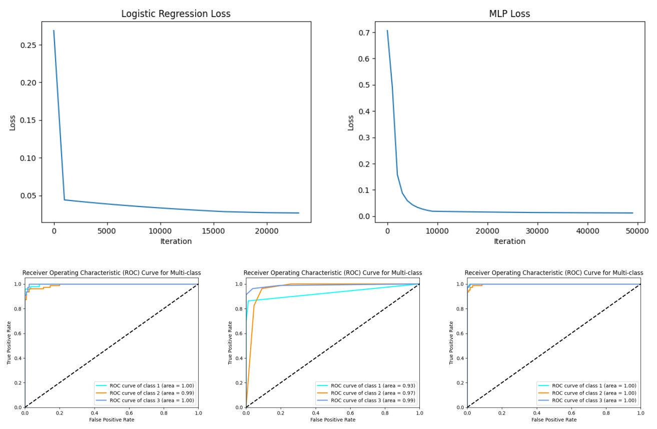

Fig. 5. Results for our models. The first row shows the variation of loss for logistic regression model (left) and multi-layer perceptron (right). The second row shows the ROC curve for the logistic regression model (left), k-nearest neighbors model (middle), and multi-layer perceptron model (right).

6. BONUS1: AN ATTEMPT: A NOVEL CLASSIFIER

6.1. Methodology

According to the doc from pytorch, we established 3 basic models based on `pytorch`. We encapsulated 3 different models: logistic regression (LE) k nearest neighbors (KNN) and multi-layer perceptron (MLP) **manually** to call better the functions in the models, which can fit our self-defined models.

We want to use the default features (that is, do not operate the ResNet18) from the ResNet18 and apply the LE, KNN, and MLP models on the default features. And the steps are the same as the 3.1 part.

6.2. Results

The training results are not as good as the ResNet18. We provide the accuracy and the time cost in Table 2 and the ROC curves in Fig. 5. Though the classification results are not so good, the time cost has been cut drastically.

6.3. Analysis

The main reason for the huge gap between our experimental results and the direct use of ResNet18 is that our machine learning mode is mainly traditional machine learning. In the training process, we only trained the traditional machine learning part, but did not make any changes to the feature extraction part of ResNet18. This results in some features not being extracted. The speed increase is also mainly due to the fact that we only trained the traditional machine learning part, resulting in the speed increase. **We will discuss the improvement together with the features extraction part** in the following section.

7. BONUS2: EXTRACTING FEATURES WITH CONVOLUTIONAL NEURON NETWORKS

7.1. Methodology

Based on pytorch, to better our own design of the classifier, we finished a convolutional neuron network (CNN) **manually**, to extract the features better. And after extracting the feature, we apply the self-defined models (provided in section 6).

Initially, we processed the images. We resize the image, adjust its brightness and contrast, and convert it to a grayscale, preserving the final gray gradient phase. Two examples of the preprocessed image are shown in Fig. 6.

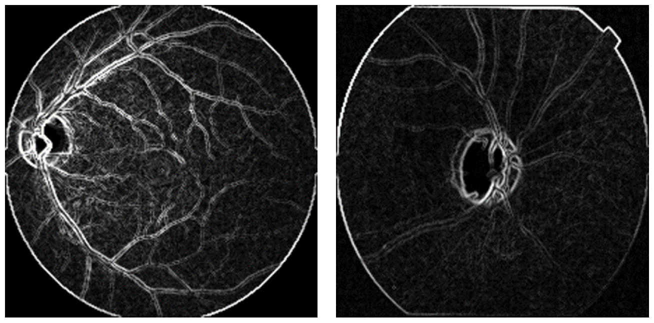

Fig. 6. Two examples of our preprocessing results. We resize each graph to be 224×224 in size (the same as the example provided by TA), adjust the image contrast to 2.5 times its original level, and reduce the brightness by 20 units. Then we convert the input image to grayscale, and apply Gaussian blur with a kernel size of (1, 1) to reduce noise and improve edge detection, and then used the Canny edge detection algorithm with thresholds of 30 and 65 to compute the gradient image.

Our design is pretty much like the classic CNN structures shown in Fig. 7, which is pretty much like that in [4]. Our strategy for CNN is as follows:

Convolution Layers

To extract features, we apply Each convolutional layer employs a kernel size of 3×3, with a stride of 1 and padding of 1.

Pooling layers Pooling layers is used for decreasing computational complexity while preserving essential feature information. In our work, each pooling layer employs a kernel

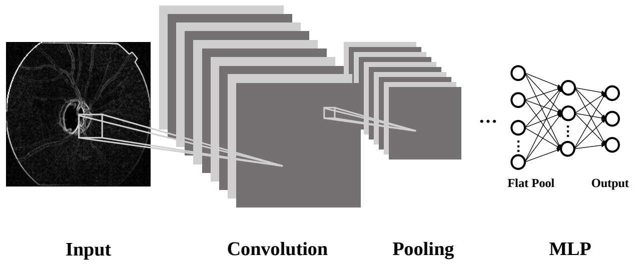

Fig. 7. Structure for our self-defined CNN model.

Model	CNN	ResNet
Accuracy	99.53%	100%
Training Time (s)	155	348

Table 3. Comparing the classification accuracy and training time between our self-defined CNN model to the ResNet18 model. Due to the simple structure of our model, the training time of our model is shorter, whose accuracy is slightly lower than the ResNet18.

size of 3×3, with a stride of 2 and padding of 0. We consider ”max pooling” for downsampling the feature maps.

Fully Connected Layers

After the last downsampling, we flatten the last layer and connect it to a multi-layer perceptron to do nonlinear multi-class classification.

And our parameters are set to be:

- Convolutional Channels: `[16, 32, 64]`
- Hidden Layers for MLP: `[128]`
- Learning Rate: `0.001`
- Momentum: `0.9`
- Epochs: `3`

7.2. Results

Using the testing data for evaluating our work. The classification results are pretty good. It reaches the accuracy of 99.53% with in 155s (provided in Table 3), which is to say, only one graph failed to be recognized by the model implemented by ourselves. The final training loss is 0.002, and the ROC curve is shown in Fig. 8

7.3. Analysis

By employing image preprocessing, defining a CNN to extract image features, and using a multi-layer perceptron (MLP) for classification, we achieved significant improvements in our classification results. Although our model's accuracy is slightly lower compared to ResNet18, our training time (155s) is less than half of that of ResNet18 (348s).

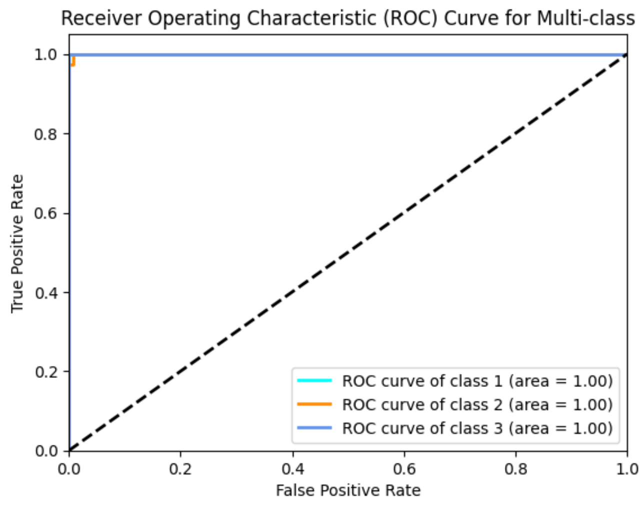

Fig. 8. The ROC curve for examining the classification results of our fully designed CNNC model.

However, our model is not without its limitations. Firstly, the simplicity of our convolutional layers renders them inadequate for handling the complexity of fundus images. Secondly, converting images to grayscale may result in information loss. Additionally, our preprocessing techniques are relatively basic, and our fully connected layers rely on the simplest form of MLP, which collectively contributes to the lower classification accuracy compared to the ResNet18 network. It is noteworthy that the faster training speed of our model can likely be attributed to its simpler architecture.

8. BONUS3: CLASSIFYING 7 CLASSES

8.1. Methodology

In this extension of our project, we modified the pre-trained ResNet18 model to adapt from a three-class to a seven-class classification system. This task was aimed at exploring the model's performance on a more complex classification scenario with a newly constructed dataset for seven distinct categories.

The dataset was meticulously organized, ensuring a balanced distribution of classes to prevent any bias during the training process. Standard image preprocessing techniques such as resizing, normalization, and augmentation were employed to enhance the model's training efficiency and robustness.

The architecture of the ResNet18 was modified by adjusting the last fully connected layer to output seven classes instead of the original three. This change was crucial to accommodate the diverse categories present in the new dataset.

The training process involved several key steps:

i. Loading the balanced seven-class dataset and applying necessary preprocessing steps.

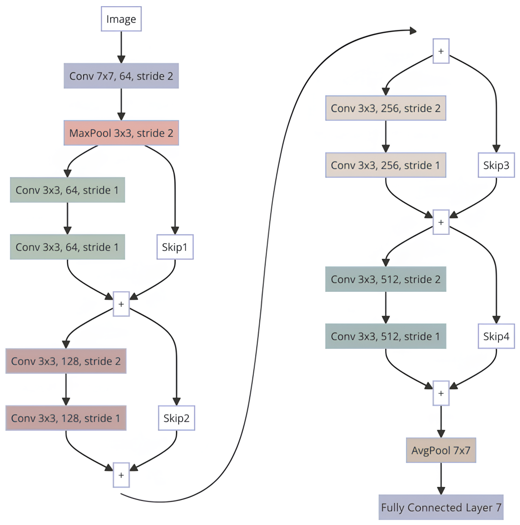

Fig. 9. The architecture of the ResNet18, modified by adjusting the last fully connected layer to output seven classes instead of the original three.

ii. Modifying the neural network's architecture to include a fully connected output layer designed for seven classes.
iii. Employing stochastic gradient descent for optimization, with a learning rate initially set to 0.001 and momentum of 0.9.
iv. Conducting the training over multiple epochs, monitoring performance on a validation set to prevent overfitting, and adjusting training parameters when necessary.

These steps ensured that the model was not only adapted to the complexity of classifying multiple categories but also optimized to achieve high accuracy and robustness in its predictions. The effectiveness of this methodology was later validated through comprehensive performance metrics, including ROC curves for each class, demonstrating the model's enhanced capability to discriminate among more nuanced categories.

8.2. Results

The evaluation of the modified ResNet18 model on the seven-class classification task was quantitatively assessed through the use of ROC curves for each class. As illustrated in the accompanying figure, the model demonstrated exceptional performance across all classes.

The results indicate that all seven classes achieved an area under the curve (AUC) of 1.00. This perfection in AUC values suggests that the model was able to distinguish between the classes with 100% accuracy, without any overlap or misclassification between the different classes. Such outcomes highlight the model's robustness and its capability to handle a more granular level of classification effectively.

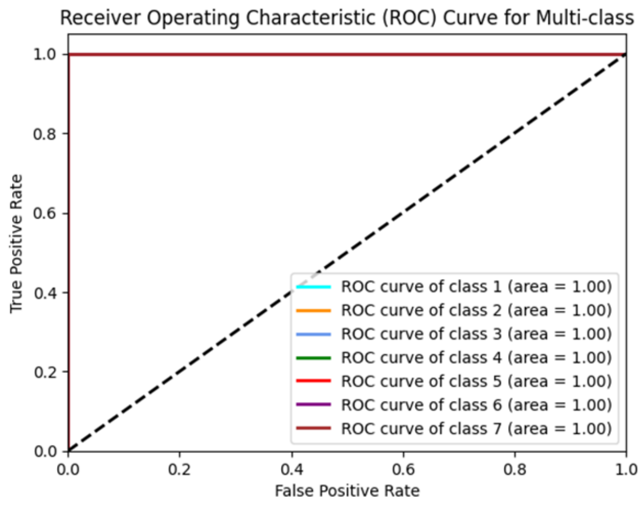

Fig. 10. ROC curves for the seven-class classification task using the modified ResNet18 model. Each curve corresponds to one of the seven classes, with each curve achieving an area under the curve (AUC) of 1.00, indicating perfect classification ability.

8.3. Analysis

The transition from a three-class to a seven-class classification system required careful consideration of the network's architecture and training dynamics. By adapting the ResNet18 model to output more classes, we observed an increase in the complexity of the feature space that the model needed to learn. This was effectively managed by optimizing training parameters and ensuring that the dataset was well-preprocessed and balanced, which was crucial for achieving high classification performance.

This experiment underscores the adaptability of convolutional neural networks to varied and complex classification tasks, further highlighting the efficacy of deep learning in medical image analysis.

9. CONCLUSION

In this project, we investigated the combination of deep learning and traditional machine learning techniques for classifying fundus lesions in medical images. By leveraging a pre-trained ResNet18 model for feature extraction and employing various traditional classifiers, we achieved promising results demonstrating this hybrid approach's effectiveness.

Our experiments showed that the ResNet18 model, both in its pre-trained and fine-tuned forms, is highly effective at extracting discriminative features from fundus images, leading to excellent classification performance. The hybrid approach of using deep learning for feature extraction and traditional machine learning for classification capitalizes on the strengths of both paradigms, resulting in a robust and interpretable system.

During our research, we designed a custom CNN for feature extraction, which provided a good balance between accuracy and training efficiency. While the custom CNN did not surpass the ResNet18 in accuracy, it highlighted the potential for creating lightweight models tailored for specific tasks.

Furthermore, we extended our classification system to handle seven classes instead of three, demonstrating that the modified ResNet18 could achieve perfect classification performance even with increased complexity. This underscores the scalability and robustness of our methodology for more nuanced classification tasks in medical image analysis.

Overall, our work shows that combining deep learning and traditional machine learning techniques can significantly enhance the accuracy and efficiency of medical image classification. Future research should focus on exploring more sophisticated models, larger and more diverse datasets, and advanced feature extraction techniques to further improve the performance and generalizability of such hybrid systems.

10. REFERENCES

[1] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun, "Deep residual learning for image recognition," *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

[2] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun, "Identity mappings in deep residual networks," *European conference on computer vision*, pp. 630–645, 2016.

[3] Kunihiko Fukushima, "Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position," *Biological cybernetics*, vol. 36, no. 4, pp. 193–202, 1980.

[4] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.