Journal of Theoretical and Applied Information Technology
15th January 2025. Vol.103. No.1

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

ADVANCED DEEP LEARNING TECHNIQUES FOR DIABETIC RETINOPATHY DETECTION USING CLAHE-GAMMA-UNSHARP HYBRID ENHANCEMENT

ARPITA NIBEDITA[1] , PRABHAT KUMAR SAHU[2] , SRIKANTA PATNAIK[3] , KUMAR SURJEET CHAUDHURY[4]

[1,2] Siksha 'O' Ansandhan (Deemed to be) University, Bhubaneswar, INDIA

[3] Interscience Institute of Management & Technology, , Bhubaneswar, INDIA

[4] KIIT Deemed to be University, Bhubaneswar, INDIA

E-mail: [1]anibedita@gmail.com, [2] prabhatsahu@soa.ac.in, [3] srikantapatnaik@iimt.ac.in, [4] surjeet.chaudhuryfcs@kiit.ac.in

ABSTRACT

Diabetic Retinopathy (DR) is a leading cause of blindness worldwide, affecting individuals of all ages. Each year, a significant number of people suffer vision loss due to DR. The diagnosis of DR requires precision, as even minor errors can lead to severe consequences. Misdiagnosis is common and can significantly increase patient morbidity. To address these challenges, this paper presents a comprehensive study on the classification of Diabetic Retinopathy (DR) images using three deep learning models: ResNet101V2, InceptionResNetV2, and a custom CNN model. The goal is to evaluate the performance of these models and improve generalization through a stacking ensemble approach with a logistic regression meta-learner. Each model was fine-tuned to enhance feature extraction and classification performance. ResNet101V2 achieved a peak training accuracy of 95.95% but exhibited overfitting, with a test accuracy of 79.81%. InceptionResNetV2 achieved an exceptional training accuracy of 99.91%, though its test accuracy was slightly better at 81.31%. The custom CNN, a simpler architecture, demonstrated a balanced performance with 87.24% training accuracy and 86.63% test accuracy, highlighting its strong generalization capabilities. A stacking ensemble was implemented to improve prediction accuracy, combining the outputs of the three models using pseudo-labeling techniques. The meta-learner enhanced the overall classification accuracy, leveraging the strengths of the individual models. This ensemble approach proved effective in improving model generalization to unseen data. Future work will focus on mitigating overfitting in more complex models, exploring transformer-based architectures, and applying these models to real-world clinical datasets for DR detection.

Keywords: *Diabetic Ratinopathy, ResNet101V2, InceptionResNetV2, Custom CNN, Stacking Ensemble, Logistic Regression, Random Forest, Gradient Boosting, Decision Tree*

1. INTRODUCTION

Diabetic Retinopathy (DR) is one of the most severe complications of diabetes, affecting the eyes and potentially leading to vision loss and blindness if not detected and treated early. It results from damage to the small blood vessels in the retina, the light-sensitive tissue at the back of the eye, caused by prolonged periods of high blood sugar [1]. The global prevalence of diabetes has sharply increased over the last few decades, making DR a significant public health concern. By 2040, it is estimated that over 642 million people will be affected by diabetes, with a substantial portion of these individuals at risk of developing DR [2].

The classification and severity of DR are determined by the extent of retinal damage, which can be divided into different stages. The four primary stages of DR include:

- Mild Non-Proliferative DR: This is the earliest stage of DR, characterized by microaneurysms, which are small areas of balloon-like swelling in the blood vessels of the retina. Patients at this stage may not experience noticeable

vision changes, but early detection is essential to prevent disease progression [3].

- Moderate Non-Proliferative DR: At this stage, the disease progresses with the blockage of some retinal blood vessels, leading to poor blood circulation in the retina. While asymptomatic for many patients, further retinal damage can cause vision disturbances, making intervention more crucial [4].
- Severe Non-Proliferative DR: As more blood vessels become blocked, the retina is deprived of blood flow. This stage is marked by significant damage to the retina, with an increased risk of the disease progressing to the proliferative stage. Patients may experience more severe vision problems as blood vessels continue to deteriorate [5].
- Proliferative DR: This is the most advanced stage, where new abnormal blood vessels grow in response to the lack of oxygen in the retina. These new vessels are fragile and can bleed into the vitreous, causing severe vision impairment or blindness. Proliferative DR can also lead to retinal detachment and other severe eye conditions, making it the most dangerous stage of the disease [6].

Given the asymptomatic nature of early DR, regular screening and early detection are crucial to preventing irreversible vision damage. Traditionally, ophthalmologists examine retinal images through fundus photography to diagnose DR, but this process is labor-intensive and prone to subjective errors. The need for automated detection and classification systems has driven the development of machine learning and deep learning techniques. By leveraging large datasets of retinal images, these techniques have shown promise in accurately identifying different stages of DR, reducing diagnostic time, and improving early intervention [7].

1.1 Problem Statement & Key Challenges

Contextual Overview Diabetic Retinopathy (DR) is a leading cause of blindness globally, requiring timely and accurate detection for effective treatment. Existing manual diagnosis processes are time-consuming, error-prone, and reliant on expert knowledge, while current automated approaches face challenges in generalization and accuracy.

Despite advancements in deep learning, individual models like ResNet and InceptionResNet often exhibit overfitting or inconsistent performance, particularly on unseen clinical datasets. This necessitates robust techniques that can generalize across varying data distributions.

In this study, we focus on enhancing the performance of deep learning models for DR classification by employing a stacking ensemble method. The individual models include ResNet101V2, InceptionResNetV2, and a custom CNN architecture, each with feature extraction and classification strengths. By combining their outputs through a logistic regression meta-learner, we aim to improve the overall accuracy and robustness of the classification system. Additionally, we use pseudo-labeling to refine the training process, leveraging the models' highest confidence predictions to guide further improvements.

2. RELATED WORK

Recent advancements in deep learning have significantly enhanced the accuracy of diabetic retinopathy (DR) detection and classification. Romero-Aroca et al.[8] validated a deep learning algorithm specifically designed for diabetic retinopathy, demonstrating its effectiveness in clinical settings.
Esteva et al.[9] extended the application of deep neural networks beyond ophthalmology by achieving dermatologist-level classification of skin cancer, indicating the potential of deep learning techniques in medical image analysis.

In a comprehensive survey, Abushawish et al.[10] explored various automatic diabetic retinopathy detection and grading systems, highlighting the evolution and comparative performance of deep learning methodologies. The use of ensemble approaches has also gained traction, as illustrated by Bodapati and Balaji [11], who proposed a self-adaptive stacking ensemble method with attention-based deep neural networks to predict DR severity effectively.

Furthermore, Enkvetchakul et al. [12] focused on data resampling and meta-learning techniques to improve the recognition of diabetic retinopathy, addressing challenges related to data imbalance.

The integration of gated-attention mechanisms in deep neural networks for DR severity classification was investigated by Bodapati et al. [13], showcasing improved classification performance. Reddy et al. [14] presented an ensemble-based machine learning model for DR classification, emphasizing the benefits of combining multiple learning algorithms. Ribeiro et al. [15] provided insights into explainable AI, crucial for gaining trust in deep learning models used in medical applications.

The exploration of deep learning architectures continued with Tymchenko et al. [16], who proposed a new approach for diabetic retinopathy detection. Samanta et al. [18] demonstrated the feasibility of automated DR detection using convolutional neural networks on a limited dataset, underscoring the importance of robust models in clinical practice. El Houby [19] applied transfer learning techniques to enhance DR stage classification, while Zang et al. [20] introduced a classification framework utilizing OCT angiography and deep learning analysis.

Moreover, Arrieta et al. [21] emphasized the potential of semi-supervised and self-supervised learning techniques in DR detection, which could address the scarcity of labeled data. Lastly, Ma et al. [22] combined transformers and convolutional neural networks to establish a joint ordinal regression and multiclass classification model for DR grading, indicating a promising direction for future research in this field.

Overall, these studies illustrate the rapid progress in applying deep learning techniques to diabetic retinopathy detection and classification, highlighting various innovative approaches and methodologies.

3. MATERIALS & METHODOLOGY

This section describes the classification approach for diabetic retinopathy (DR) using the APTOS dataset, illustrated in Fig. 1. The process starts with preprocessing the retinal images, followed by data augmentation to expand the dataset and support effective training of deep convolutional neural networks (CNNs). The method employs transfer learning with pre-trained DenseNet and ResNet models, which are fine-tuned for feature extraction and classification. DenseNet and ResNet are chosen for their strong performance and efficiency, ensuring accurate DR classification with minimal computational load.

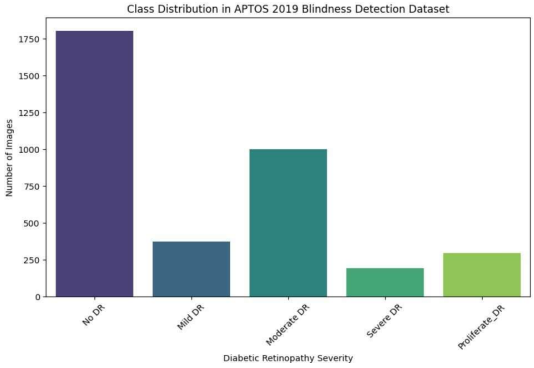

Figure 1: Diabetic Retinopathy Class Distribution

3.1 Dataset Overview

The APTOS 2019 Blindness Detection Dataset is employed to train and evaluate machine learning models for detecting Diabetic Retinopathy (DR) from retinal fundus images. Provided by the Asia Pacific Tele-Ophthalmology Society (APTOS) as part of a Kaggle competition, this dataset contains high-resolution retinal images categorized into five classes based on DR severity as shown in table 1 and figure 1. For this research, the" Diabetic Retinopathy 224x224 Gaussian Filtered" dataset was used, where images were resized to 224x224 pixels and processed with a Gaussian filter to enhance significant features and reduce noise. This preprocessing is particularly beneficial for deep learning models like CNNs, enabling them to focus on critical patterns for accurate DR detection, which is vital for early diagnosis and treatment.

Initially, the dataset is divided into five classes:

- No DR (0): No diabetic retinopathy.
- Mild (1): Mild non-proliferative diabetic retinopathy.
- Moderate (2): Moderate non-proliferative diabetic retinopathy.
- ProliferateDR (3): Proliferative diabetic retinopathy
- Severe (4): Non-proliferative diabetic retinopathy

Table 1: Number of Images in Each Class of the APTOS 2019 Blindness Detection Dataset

Class	Label	Number of Images
No_DR	0	1805
Mild	1	370
Moderate	2	999
Severe	3	193
Proliferative_DR	4	295
Total		**3662**

Table 2: Grouped Classes for Binary Classification in the Diabetic Retinopathy Dataset

Grouped Class	Original Classes Combined	Number of Images
No_DR	No_DR, Mild, Moderate	3174
Severe	Proliferative_DR, Severe	488
Total		**3662**

Table 1 summarizes the distribution of images in each class.

For binary classification, these five classes were consolidated into two groups. The first group, labeled No DR, merges the "No DR", "Mild", and "Moderate" classes, representing less- severe cases as shown in table 2. The second group, labeled Severe, combines the "Proliferative DR" and "Severe" classes, representing more advanced stages of the disease. This grouping helps in distinguishing between cases that require immediate medical intervention and those that do not.

3.2 Data Preprocessing

The objective of preprocessing the APTOS DR dataset is to enhance the quality and consistency of the image data, ensuring it is suitable for accurate and reliable model training and evaluation. This involves cleaning the dataset by removing corrupt images, augmenting data to increase variability and prevent overfitting, normalizing and resizing images for uniformity, and addressing class imbalances to avoid biased predictions. Additionally, feature extraction and image enhancement techniques are applied to improve the visibility and representation of key features, ultimately aiming to optimize the performance of classification models in diagnosing the severity of diabetic retinopathy. Data preprocessing is critical in preparing the APTOS Diabetic Retinopathy (DR) dataset for practical analysis and modeling.

3.2.1 CLAHE-Gamma- Unsharp Hybrid Enhancement (CGUHE)

We have proposed a three-step image enhancement approach called as CCUHE using various techniques: CLAHE (Contrast Limited Adaptive Histogram Equalization), Gamma Correction, and Unsharp Masking. Below is a theoretical and mathematical description of each function:

- ***CLAHE Enhancement***

CLAHE is designed to improve the contrast of images, particularly in low-contrast areas, by applying histogram equalization to small tiles of the image and then combining them. This prevents over-amplification of noise in homogeneous regions and improves local contrast.

CLAHE operates as follows:

a) Histogram Equalization: It redistributes the intensity values of the pixels to enhance contrast.

b) Clip Limit: The histogram's cumulative distribution function is clipped at a specified clip limit to avoid excessive contrast enhancement.

c) Tile Grid Size: The image is divided into small tiles (e.g., 8x8), and equalization is applied to each tile.

For each tile:

- Compute the histogram of pixel intensities.

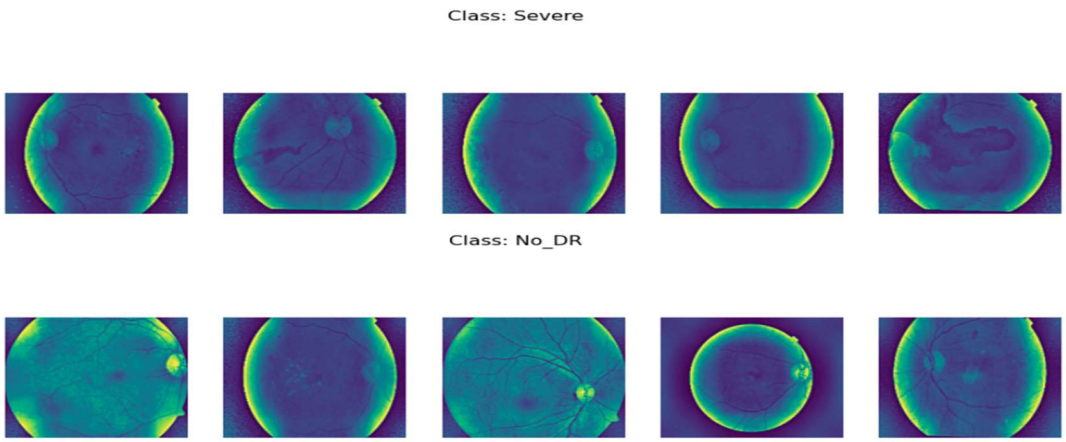

Figure 2 DR Images after CGUHE

- A
  pply the clipping limit to the histogram to avoid over-enhancement.
- Use linear interpolation to map the pixel values to the new histograms se defined in the equation 1:

$$\boldsymbol{NewIntensity}\,(\boldsymbol{x},\boldsymbol{y}) = \boldsymbol{CDF}\,(\boldsymbol{Intensity}(\boldsymbol{x},\boldsymbol{y})) \times \boldsymbol{(maxIntensity\text{-}minIntensity)+minIntensity} \quad (1)$$

where CDF is the cumulative distribution function of the pixel intensity.

- ***Gamma Correction***

Gamma Correction adjusts the brightness of an image by applying a non-linear transformation to pixel values. It is used to correct the brightness and contrast based on the gamma parameter.
The equation 2 defines gamma correction:

$$I_{\text{corrected}} = 255 \times \left(\frac{I_{\text{original}}}{255}\right)^{\frac{1}{\gamma}} \quad (2)$$

where:

- I_{original} is the original pixel intensity value.
- $I_{\text{corrected}}$ is the gamma-corrected pixel intensity value.
- γ is the gamma value. For $\gamma > 1$, the image appears darker; for $\gamma < 1$, it appears lighter.

- ***Unsharp Masking***

Unsharp Masking is a technique used to enhance image sharpness by emphasizing edges. It involves subtracting a blurred version of the image from the original image to enhance the high-frequency details.

Unsharp Masking defines in equation 3 involves the following steps:

a) Gaussian Blurring: Apply a Gaussian filter to the image to create a blurred version. The Gaussian kernel's size and standard deviation control the amount of blurring.
b) Subtracting the Blurred Image: Calculate the difference between the original and blurred images to obtain the mask.
c) Adding the Mask: Add a weighted mask version to the original image to enhance the edges.

$$I_{\text{unsharp}} = I_{\text{original}} + \alpha \times \left(I_{\text{original}} - I_{\text{blurred}}\right) \quad (3)$$

where:

- I_{original} is the original image.
- I_{blured} is the image after Gaussian blurring.
- α is a weight factor that controls the strength of enhancement.

Summary of the CCUHE

- CLAHE: Improves local contrast.
- Gamma Correction: Adjusts overall brightness.

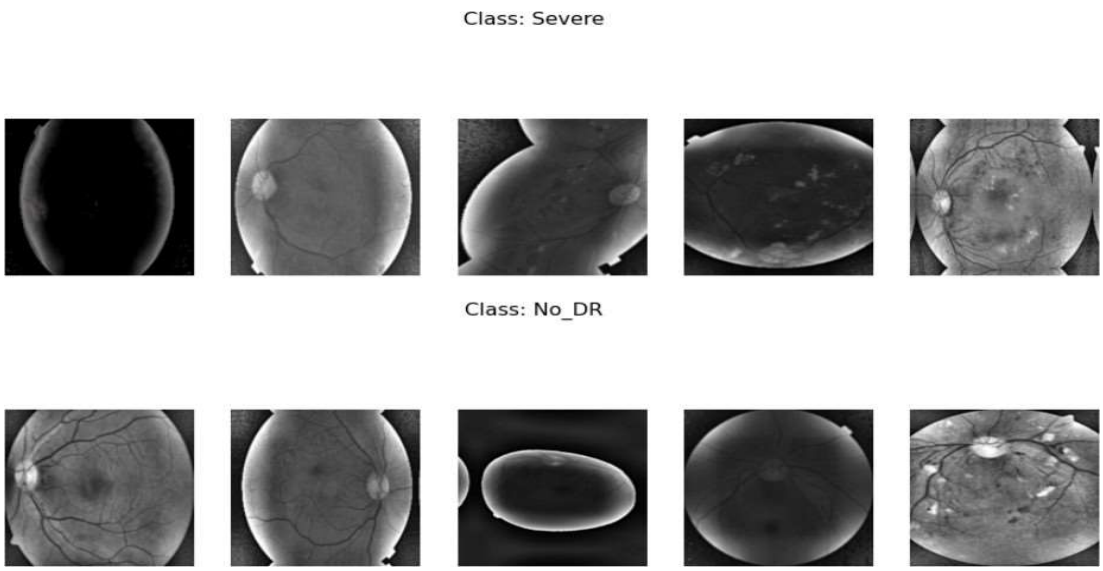

Figure 3: DR Images after Augmentation

- Unsharp Masking: Enhances edges and sharpness. Together, these methods enhance the visibility of features in the image, making them more suitable for further analysis or processing. This hybrid proposed CLAHE is used to improve local contrast, Gamma Correction to adjust brightness levels adaptively, and Unsharp Masking to enhance edge details. By combining these techniques, CGUHE improves image quality, ensuring better feature visibility and sharper details. Sample images after applying this hybrid approach are displayed in Figure 2 for visual reference.

3.3 A Unique Data Augmentation Method

The augmentation process aims to enhance the diversity and variability of the training dataset for diabetic retinopathy classification by applying a series of geometric and photometric transformations. These transformations, including rotations, flips, brightness and contrast adjustments, blurring, color modifications, and elastic distortions, help simulate real-world variations in retinal
Images, such as changes in camera angle, lighting conditions, and image quality. By augmenting the data in this way, the process aims to improve the model's ability to generalize across different imaging scenarios, reducing overfitting and increasing the model's robustness and accuracy when classifying the different stages of diabetic retinopathy. This approach ensures that the model is exposed to a wide range of visual patterns, enhancing its capacity to effectively recognize relevant features across varied images. The given augmentation pipeline uses the Albumentations library to perform a series of image transformations to increase the diversity of the training dataset, thereby helping to improve the robustness and generalization ability of machine learning models for tasks such as diabetic retinopathy classification.

The steps are as follows:

a) Randomrotate90(p=0.5)

Description: Randomly rotates the image by 90 degrees, either 0, 90, 180, or 270 degrees.
A transformation matrix R for 90-rotation can be expressed as degree:

$$\begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix}$$

This operation is applied to the image pixels with a probability $p = 0.5$, meaning it occurs for 50% of the images.

b) Horizontal Flip(p=0.5)

Description: Flips the image horizontally with a probability of 0.5.
If $I(x, y)$ represents the intensity of the image at position (x, y), the horizontally flipped image $I'(x, y)$ is defined in equation 4 as:

$$I'(x, y) = I(width - x - 1, y) \quad (4)$$

This mirrors the image along the vertical axis.

c) Verticalflip (p=0.5)

Description: Flips the image vertically with a probability of 0.5.

The vertically flipped image $I'(x, y)$ is defined by the equation 5 as :

$$I'(x, y) = I(x, \text{height} - y - 1) \quad (5)$$

This mirrors the image along the horizontal axis.

d) Randombrightnesscontrast (p=0.5)

Adjusts the brightness and contrast of the image.

- Brightness: Modifies each pixel value $I(x, y)$ by adding a random value ΔB (brightness shift) defined by the equation 6 as:

$$I'(x, y) = I(x, y) + \Delta B \quad (6)$$

- Contrast: Scales the pixel value relative to the mean intensity of the image:

$$I'(x, y) = \alpha \times (I(x, y) - \mu) + \mu \quad (7)$$

where α is a random contrast factor and μ is the mean intensity.

e) GaussianBlur(blurlimit=3, p=0.2)

Applies a Gaussian blur to the image, with a small kernel size limit of 3.

- The Gaussian blur is achieved by convolving the image with a

Gaussian kernel $G(x, y)$:

$$I'(x, y) = I(x, y) * G(x, y)$$

where:

$$G(x, y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+\rho^2}{2\sigma^2}}$$

This reduces noise and smoothens the image.

f) HueSaturationvaliue(hue_shift_limit=10 ,sat_shift_limit=15,val_shift limit=10, p=0.3)

Adjusts the image's hue, saturation, and value (HSV).

- Hue shift: adds a random shift Δh to the hue component.
- Saturation shift: scales the saturation component by $s + \Delta s$.
- Value shift: adjusts the brightness value by adding Δv.

g) Resize (height=256, width=256, p=1.0)

Resize the image to a fixed size of 256x256 pixels to ensure a uniform input size for the model. Interpolation techniques such as bilinear or bicubic interpolation are typically used to resize the image.

h) ElasticTransform(alpha=1.0, sigma=50.0, alpha affine=50.0, p=0.3)

Applies elastic deformation to the image, simulating random movements or warping effects.

Elastic transformation is defined by applying a displacement field generated by random Gaussian distributions, which is determined by equation 8 as:

$$Displacement(x, y) = (\Delta x, \Delta y) \quad (8)$$

Where Δx and Δy are smoothed with Gaussian filters of standard deviation σ.

i) GridDistortion(num steps=5, distort limit=0.3, p=0.2)

Warps the image by shifting pixel locations along a grid.

The grid is distorted by moving its points randomly within a specified distortion limit. This generates a new mapping function that displaces the pixels.

j) OpticalDistortion(distor_limit=0.3, shift_limit=0.2, p=0.2)

Simulates lens or camera optical distortions, such as barrel or pincushion effects.

Adjusts pixel positions using a distortion function defined by parameters k_1 and k_2 that control the intensity of the effect.

k) Solarize (threshold=128, p=0.1)

Description: Inverts the pixels' intensity above a certain threshold, creating a solarized effect defined by equation 9.

$$I'(x, y) = \begin{cases} I(x, y), \text{ if } I(x, y) < \text{threshold}, \\ 255 - I(x, y), \text{ otherwise.} \end{cases} \quad (9)$$

This effect is often used to highlight certain features in the image. Using this unique

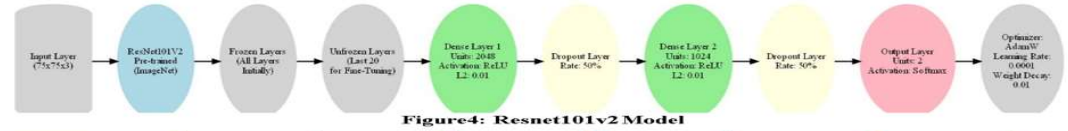

Figure4: Resnet101v2 Model

Figure 5: Resnet101v2 Model

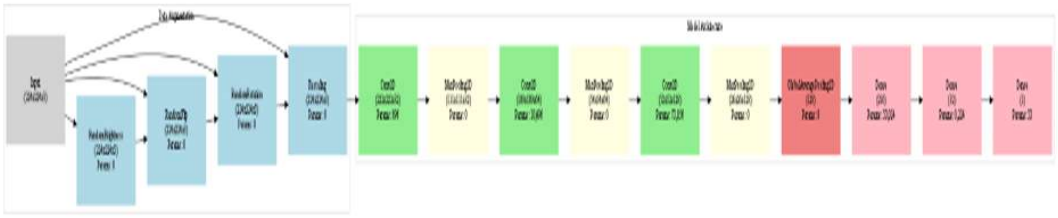

Figure 6: Custom CNN Model

augmentation pipeline, the first function applies predefined transformations to an input image.

The second function traverses a source folder containing images, processes each image using the augmentation above function, and saves the processed images to a destination folder while maintaining the original directory structure.

3.4 Data Loading and Splitting Summary

The augmented images with two classes are divided into training and validation subsets by reserving 20% of the images from the training directory for validation. The training dataset contains most images, while the validation dataset is a smaller subset used to tune the model. The images are resized to 75x75 pixels and grouped into batches of 32. The model is configured to use binary cross-entropy as the loss function, indicating a binary classification task. The test dataset is loaded separately from the testing directory and is used to evaluate the model's performance after training. The code reports the number of images found in each class and the number used for training, validation, and testing. The code loads images from specified directories and divides them into training, validation, and test datasets. The training dataset contains 2,928 images from two classes, with 2,343 images used for training and 585 reserved for validation. The test dataset is loaded separately, containing 733 images. All images are resized to 75x75 pixels and grouped into batches of 32. The binary cross-entropy loss function is used, indicating this is a binary classification task. The output confirms the number of images found in each class and the number used for training (2,343 images), validation (585 images), and testing (733 images).

4.MODEL IMPLEMENTATION

Implementing different models like ResNet, InceptionNet, and others aims to conduct a comparative analysis to identify the most influential architecture for a specific image classification task. This involves evaluating each model's accuracy, training efficiency, generalization capability, and robustness to overfitting. By experimenting with various architectures, the goal is to optimize the model for the given dataset and task, ultimately selecting the model that best balances these factors for improved predictive performance.

4.1 Resnet101v2 Model

The MirroredStrategy enables synchronized training, defining the image dimensions as 75x75 pixels for binary classification. Within this strategy, the ResNet101V2 model is loaded with ImageNet weights, excluding the top layer. Initially, all layers are frozen except for the last 20, unfrozen for fine-tuning. The model architecture includes ResNet101V2 as the base, followed by a GlobalAveragePooling2D layer, a dense layer with 2048 units and ReLU activation, L2 regularization, batch normalization, and dropout. An additional dense layer with 1024 units is included, along with L2 regularization, followed by batch normalization and dropout. The final output layer consists of two units with a softmax activation function. Callbacks such as ModelCheckpoint, EarlyStopping, ReduceLROnPlateau, and LearningRateScheduler are configured to

improve training efficiency and model performance. The AdamW optimizer is used with a learning rate of 0.0001 and weight decay of 0.01, and the model is compiled with sparse categorical cross-entropy loss appropriate for the classification task. The model summary is displayed to verify the architecture before training [23][24].

4.2 Inceptionresnetv2 Model

The MirroredStrategy enables synchronized training, defining the image dimensions as 75x75 pixels for binary classification. The MirroredStrategy enables synchronized training within this strategy, defining the image dimensions as 75x75 pixels for binary classification. The inceptionresnetv2 model is loaded with ImageNet weights within this strategy, excluding the top layer. Initially, all layers are frozen except for the last 20, unfrozen for fine-tuning. The model architecture includes InceptionResNetV2 as the base, followed by a GlobalAveragePooling2d layer, a dense layer with 2048 units and ReLU activation, L2 regularization, batch normalization, and dropout. An additional dense layer with 1024 units is included, along with L2 regularization, followed by batch normalization and dropout. The final output layer consists of two units with a softmax activation function. Callbacks such as ModelCheckpoint EarlyStopping, ReduceLROnPlateau, and LearningRateScheduler are configured to improve training efficiency and model performance. The Adam optimizer is used with a learning rate of 0.0001 and weight decay of 0.01, and the model is compiled with sparse categorical cross-entropy loss appropriate for the classification task. The model summary is displayed to verify the architecture before training. Is loaded with ImageNet weights, excluding the top layer. Initially, all layers are frozen except for the last 20, unfrozen for fine-tuning. The model architecture includes InceptionResNetV2 as the base, followed by a GlobalAveragePooling2d layer, a dense layer with 2048 units and ReLU activation, L2 regularization, batch normalization, and dropout[25, 26].

An additional dense layer with 1024 units is included, along with L2 regularization, followed by batch normalization and dropout. The final output layer consists of two units with a softmax activation function. Callbacks such as ModelCheckpoint, EarlyStopping, ReduceLROnPlateau, and LearningRateScheduler are configured to improve training efficiency and model performance. The Adam optimizer is used with a learning rate of 0.0001 and weight decay of 0.01, and the model is compiled with sparse categorical cross-entropy loss appropriate for the classification task. The model summary is displayed to verify the architecture before training.

4.3 Custom CNN model

The new, simpler CNN model for binary classification is designed with an input image dimension of 75x75 pixels and consists of multiple convolutional layers with increasing filter sizes to capture different levels of features. The architecture begins with a rescaling layer that normalizes the pixel values, followed by a series of convolutional layers: 32, 64, 128, 256, and 512 filters, each with a kernel size of 3x3, ReLU activation, and padding set to "same" to preserve the spatial dimensions. Maxpooling2d layers are included after each convolutional block to downsample the feature maps, and dropout layers are added to prevent overfitting by randomly dropping connections. The model is flattened and followed by a fully connected dense layer with 1024 units and ReLU activation, concluding with an output layer of two units using softmax activation to produce the class probabilities. Callbacks such as ModelCheckpoint (to save the best model weights), EarlyStopping (to halt training when validation loss stops improving), ReduceLROnPlateau (to reduce the learning rate when validation loss plateaus), and LearningRateScheduler (to schedule the learning rate decay per epoch) are used to optimize and improve training performance. The model is compiled using the Adam optimizer with a learning rate 0.0001, using sparse categorical cross-entropy as the loss function for the classification task and accuracy as the evaluation metric.

5. EVALUATION MATRIX

To evaluate the performance of models or methods, several key metrics are commonly used:

a) Accuracy

Accuracy measures the proportion of correctly classified instances out of the total number of instances. It is calculated as:

$$\text{Accuracy} = \frac{\text{True Positives (TP)} + \text{True Negatives (TN)}}{\text{Total Instances}}$$

Row	ResNet101V2_pred	InceptionResNetV2_pred	Custom_Model_pred	Pseudo Label
0	0.985774	0.206377	0.955344	0
1	0.014226	0.793623	0.044656	1
2	0.948782	0.984780	0.971561	1
3	0.051218	0.015220	0.028440	0
4	0.322395	0.999744	0.922757	1

Table 3: Sample Predictions from Different Models

b) Precision

Precision quantifies the number of true positive predictions divided by the total number of positive predictions made by the model. It indicates the quality of positive predictions. It is given by:

$$\text{Precision} = \frac{\text{True Positives (TP)}}{\text{True Positives (TP)} + \text{False Positives (FP)}}$$

c) Recall

Recall measures the number of true positives divided by the total number of actual positives. It reflects the model's ability to identify all relevant instances. The formula for recall is:

$$\text{Recall} = \frac{\text{True Positives (TP)}}{\text{True Positives (TP)} + \text{False Negatives (FN)}}$$

d) F1 Score

The F1 Score is the harmonic mean of precision and recall. It provides a single metric that balances precision and recall, which is especially useful for imbalanced datasets. It is calculated as:

$$\text{F1 Score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

e) AUC-ROC

The AUC-ROC is a performance measurement for classification problems at various threshold settings. It represents the area under the ROC curve, which plots the True Positive Rate (Recall) against the False Positive Rate across different thresholds. The AUC value ranges from 0 to 1, where a higher value indicates better model performance.

5.1 Experiments and Performance Evaluation

This section details the experiments conducted with the ResNet101V2 model, including the fine-tuning process and its performance evaluation. The ResNet101V2 model, initially trained with all layers frozen, had the last 20 layers unfrozen for fine-tuning. This approach aimed to enhance the model's ability to adapt to specific dataset features. During the final training epochs, the model achieved a peak training accuracy of 95.95% at epoch 96, with a corresponding validation accuracy of 80.82%. The learning rate was systematically reduced throughout the training process to improve model convergence, with the final learning rate dropping to approximately 6.23e-07 by epoch 100.

Despite these adjustments and high training performance, the model's test accuracy was slightly lower at 79.81%, suggesting that the model may be experiencing some overfitting or that further refinement is needed to achieve better generalization. The test loss of 7.302 corroborates this observation, indicating that while the model performs well on training and validation sets, further fine-tuning or additional strategies might be required to optimize performance and reduce the generalization gap.

The InceptionResNetV2 model was fine-tuned with a specific configuration for the last 20 layers, while the first two layers were explicitly frozen to maintain pre-trained feature extraction capabilities. The training process extended to 100 epochs, during which the model demonstrated an impressive training accuracy of up to 99.91% and a validation accuracy of 78.77%. Throughout the final epochs, the learning rate was progressively reduced, aiding in the convergence of the model. Despite these high training accuracies, the model's test accuracy was 81.31%, with

a test loss of 3.383. This suggests that while the model achieved excellent performance on the training and validation sets, there is a slight gap in its generalization to unseen data, indicating potential areas for further improvement or fine-tuning to enhance its overall performance. The custom CNN model, comprising several convolutional layers and

dropout for regularization, was trained over 50 epochs. During training, the model showed an accuracy of up to 87.24% and achieved a peak validation accuracy of 89.73%. Despite these high-performance metrics, the model's validation loss did not improve beyond a certain point, reducing the learning rate as part of the learning rate scheduler's strategy. The model's test performance, with a final accuracy of 86.63% and a test loss of 0.348, indicates strong generalization to unseen data. However, the model exhibited some fluctuations in validation performance, suggesting that further tuning or adjustments might be needed to optimize stability and accuracy across different data splits. In comparing the three models, the simpler CNN model stands out for its balance between training and generalization, evidenced by its robust test accuracy and lower test loss. The ResNet101V2 and InceptionResNetV2 models, while showing exceptional training performance, struggled with generalization, as indicated by their lower validation and test accuracies. This highlights the importance of model complexity and tuning in achieving optimal performance, where simpler models may sometimes provide better generalization despite having fewer parameters and less complexity compared to more advanced architectures. The pseudo-labels for training and testing datasets were generated using the prediction scores from ResNet101V2, InceptionResNetV2, and a custom model. The prediction scores from these models are stored in A data structures, which are likely pandas DataFrames for each data point as shown in table 3. The function argmax() is applied along each row (axis=1) of these DataFrames to find the index of the highest prediction score. This index represents the model with the highest confidence for that specific data point and is used to assign a pseudo-label. For instance, if the highest score is in the first column (ResNet101V2), the pseudo-label will be 0; if it is in the second column (InceptionResNetV2), the pseudo-label will be 1, and if it is in the third column (Custom Model), the pseudo-label will be 2. The purpose of generating these pseudo-labels is to create a consensus-based or ensemble learning approach where the strengths of multiple models can be leveraged for better overall performance. Using the model with the highest prediction confidence for each data point helps generate a robust training dataset
that can be used for further meta-classification tasks or ensemble learning. This technique ensures that the pseudo-labels

6. STACKING ENSEMBLE WITH META-LEARNER FOR ENHANCED MODEL PREDICTIONS

Our first objective is to consolidate the predictions from three different models—ResNet101V2, InceptionResNetV2, and a custom CNN model—into a single data frame and save it to a CSV file as shown in table 5. Then, the minimum length among the flattened prediction arrays of the three models is calculated using the min() function and ravel() method. This step is necessary to ensure that all arrays are the same size, preventing mismatches or misalignments when combining them into a single data frame. Each model's prediction array is then sliced to this minimum length, ensuring that all arrays are truncated to an equal size for uniformity. After slicing, a pandas data frame named is created with three columns corresponding to each model's predictions. This data frame is then saved as a CSV file without an index column, allowing easy comparison, sharing, and further analysis. Finally, the length of the custom model's prediction array is used to verify
that the array's length matches the determined minimum, confirming that the truncation was correctly performed. This process ensures a standardized format for evaluating the models' predictions side-by-side.

Classifier	Precision	Recall	F1-Score	Accuracy
Random Forest	0.937	0.933	0.933	0.94
Decision Tree	0.890	0.890	0.890	0.89
Gradient Boosting	0.920	0.923	0.922	0.92

Table 4: Classification Report

In the stacking ensemble approach described, the model integrates predictions from multiple base models to generate final predictions through a meta-learner. Initially, the ensemble comprises three models: ResNet101V2, InceptionResNetV2, and a custom model. These models produce individual predictions, which are then processed to ensure uniformity in length. The projections are then consolidated into a data frame, each column representing the predictions from one of the base models. This data frame is saved to a CSV file for further use. To enhance predictive performance, pseudo-labels are generated based on the base models' prediction scores. These pseudo-labels are derived by selecting the class with the highest score for each sample. This process helps in creating a robust representation of the underlying data distribution. A logistic regression model is then employed as the meta-learner. The meta-learner is trained using the pseudo-labels obtained from the training set predictions and is evaluated on the test set predictions. The performance of the stacking ensemble is assessed by the accuracy of the logistic regression model on the test data. This evaluation provides insight into how well the meta-learner can combine the predictions from the base models to make accurate final predictions. The accuracy score reflects the effectiveness of the ensemble approach in leveraging the strengths of each base model, ultimately enhancing the overall predictive capability of the ensemble.

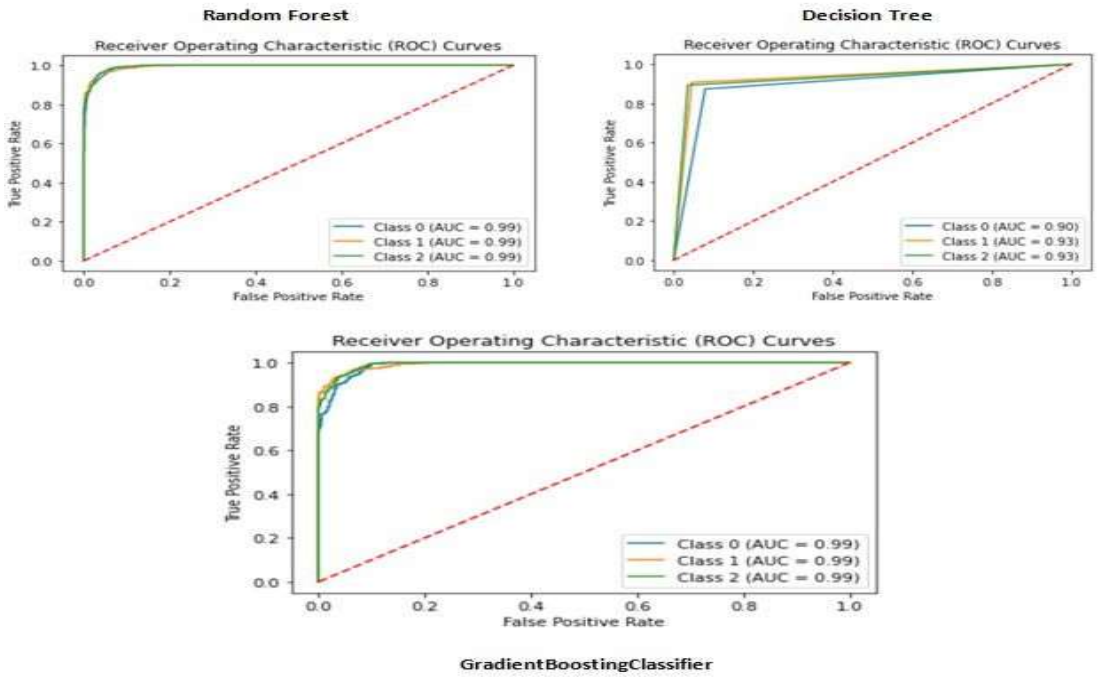

Figure 7: AUC-ROC Curves for all the Models

6.1 Results & Discussion

The performance of the Random Forest model can be analyzed using the confusion matrix, ROC (Receiver Operating Characteristic) curves, and the classification report. The confusion

matrix summarizes the model's predictions for each class. For Class 0, the model correctly predicted 407 instances while misclassifying 22 as Class 1 and 14 as Class
2. For Class 1, there were 574 correct predictions, 15 were misclassified as Class 0, and 13 as Class 2. For Class 2, the model correctly predicted 393 instances but misclassified 22 as Class 0 and 6 as Class 1. The confusion matrix reveals that while the model performs well, there are misclassifications between the classes. The ROC curves provide a more in-depth look into the model's ability to distinguish between different classes. Each curve represents one of the classes, showing the trade-off between the actual positive rate (sensitivity) and the false positive rate (1-specificity). As shown in fig. 7, the Area Under the Curve (AUC) values for all three classes are 0.99, indicating the model's excellent discriminatory power. An AUC close to 1 suggests that the model is very good at distinguishing between each class's positive and negative instances, reinforcing the model's strong classification capabilities. The classification report presents each class's precision, recall, and F1-score metrics, offering a comprehensive view of the model's performance. For Class 0, the precision, recall, and F1-score are all 0.92, indicating balanced performance. For Class 1, the model achieves a higher accuracy, recall, and F1-score of 0.95, showing the best performance among the three classes. For Class 2, the precision is 0.94, recall is 0.93, and F1-score is 0.93, suggesting strong performance but slightly lower than Class 1. The model's overall accuracy is 94%, with macro and weighted averages for precision, recall, and F1-score all at 0.94. This consistent performance across all classes demonstrates that the Random Forest model, trained with 100 decision trees, is highly effective in classifying the given dataset, as further supported by the ROC curves and confusion matrix.

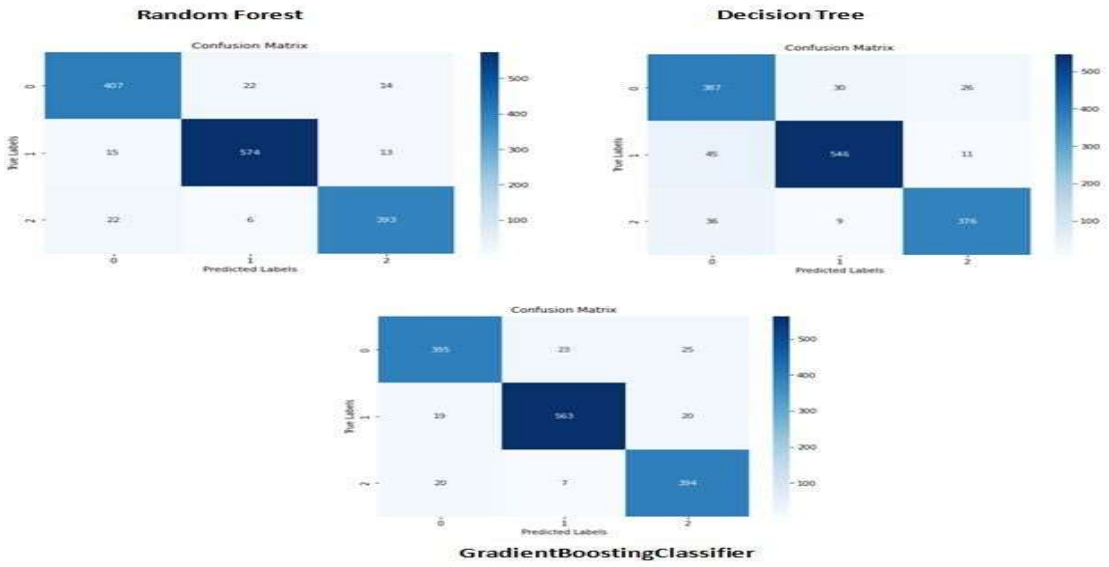

Figure 8: Confusion Matrices for different models

The training and evaluation of a decision tree classifier model are done using the scikit-learn library. The model is initialized with a specified random state for reproducibility. It is then trained on a dataset with features and corresponding pseudo labels for a multi-class classification problem. After training, the model's performance is evaluated on a test set by predicting the labels of the test data. The model's accuracy on the test data is computed by comparing the predicted and actual labels. The output shows a test accuracy of approximately 89.29%, indicating that the Decision Tree model performs reasonably on the test set.

The confusion matrix and ROC (Receiver Operating Characteristic) curves provide deeper insights into the model's performance across different classes. The confusion matrix shown in fig.8 reveals that for Class 0, the model correctly predicted 387 instances, but there were 30 misclassifications as Class 1 and 26 as Class 2. For Class 1, the model correctly predicted 546 cases, while 45 were misclassified as Class 0 and 11 as Class 2. For Class 2, instances were correctly predicted, but 36 were

misclassified as Class 0 and 9 as Class 1. The ROC curves for each class display an Area Under the Curve (AUC) of 0.90 for Class 0 and 0.93 for both Class 1 and Class 2, suggesting good but not perfect discrimination between the classes. The classification report further breaks down the model's performance using precision, recall, and F1-score metrics. For Class 0, the precision is 0.83, recall is 0.87, and the F1-score is 0.85, indicating a moderate level of classification accuracy. For Class 1, the model achieves a higher precision of 0.93, recall of 0.91, and F1-score of 0.92, demonstrating strong performance. For Class 2, the precision, recall, and F1-score are 0.91, 0.89, and 0.90, respectively, showing balanced performance but slightly lower than Class 1. The model's overall accuracy is 89%, with macro and weighted averages for precision, recall, and F1-score all at 0.89. These metrics suggest that the Decision Tree Classifier is generally effective but could benefit from further tuning or using more complex models to enhance performance. In this experiment, a Gradient Boosting Classifier (GBM) is implemented using the scikit-learn library to classify a dataset into three classes. The GBM model is initialized with a fixed random state for reproducibility. The model is trained on a dataset containing training features and pseudo labels. After training, the model's performance is evaluated on a test set, where the predicted labels are compared against the actual labels to compute the model's accuracy. The reported test accuracy for this classifier is approximately 92.22%, indicating a high level of correctness in the model's predictions. The classification performance of the Gradient Boosting Classifier is further analyzed through various metrics such as precision, recall, and F1-score, which provide a deeper understanding of how well the model distinguishes between the three classes. For Class 0, the precision is 0.91, the recall is 0.89, and the F1 score is 0.90. For Class 1, the precision is 0.95, the recall is 0.94, and the F1 score is 0.94. For Class 2, the precision is 0.90, the recall is 0.94, and the F1 score is 0.92. The model's overall accuracy is 92%, with macro and weighted averages for precision, recall, and F1-score hovering around 0.92. This demonstrates that the model performs consistently well across all classes, with a slight variance in recall and precision for each class.

A confusion matrix and Receiver Operating Characteristic (ROC) curves are plotted to further visualize the model's performance. The confusion matrix reveals that the classifier has relatively few misclassifications across all three classes, as evidenced by the high values along the diagonal and lower off-diagonal values. For instance, Class 0 has 395 correct predictions and 23 and 25 incorrect predictions as Classes 1 and 2, respectively. The ROC curves for each class show an Area Under the Curve (AUC) of 0.99, indicating the classifier's excellent discrimination ability. A nearly perfect AUC value signifies that the model effectively distinguishes between the classes with minimal error.

Table 4 presents the averaged performance metrics—precision, recall, f1-score, and accuracy—of three classifiers: Random Forest, Decision Tree, and Gradient Boosting. Among them, the Random Forest classifier achieves the highest performance with an average precision of 0.937, recall of 0.933, f1-score of 0.933, and an accuracy of 0.94. The Gradient Boosting classifier also shows strong results with an average precision of 0.920, recall of 0.923, f1-score of 0.922, and accuracy of 0.92, making it a competitive alternative. In contrast, the Decision Tree classifier has the lowest performance across all metrics, with averages of 0.890 for precision, recall, f1-score and an accuracy of 0.89. These findings suggest that ensemble methods like Random Forest and Gradient Boosting are more effective for classification tasks than single-tree models.

For instance, Bodapati et al. [11] proposed a self-adaptive stacking ensemble with attention-based mechanisms, while Ma et al. [22] introduced a hybrid model combining transformers and CNNs. Unlike these approaches, our work focuses on integrating ResNet101V2, InceptionResNetV2, and a custom CNN into a stacking ensemble, employing pseudo-labeling to further refine model predictions. This technique demonstrated improved accuracy and robustness across unseen data, addressing overfitting issues commonly observed in individual models. By comparing our results to previously published methodologies, we highlight the significant performance improvements achieved by our ensemble model, particularly in terms of generalization to diverse datasets.

7. CONCLUSION AND FUTURE WORK

In this study, we explored the performance of three deep learning architectures—ResNet101V2, InceptionResNetV2, and a custom CNN model—for Diabetic Retinopathy (DR) classification. While individual models achieved high training

accuracies, they struggled with generalization on unseen data due to overfitting. The custom CNN demonstrated more balanced performance between training and testing, suggesting its potential for practical applications. To address these limitations, we proposed a stacking ensemble approach with a logistic regression meta-learner, which leveraged the complementary strengths of the base models. The addition of a pseudo-labeling strategy further improved the ensemble's robustness, resulting in higher classification accuracy and better handling of unseen data. The novelty of this work lies in integrating pseudo-labeling with an ensemble learning framework, effectively reducing overfitting and enhancing model generalization. This innovation has significant implications for clinical applications, as it improves the reliability and accuracy of automated DR diagnosis. The proposed approach addresses a critical barrier to deploying deep learning systems in real-world medical scenarios by enabling robust performance across varying datasets.

The impact of this study extends beyond DR classification, showcasing the potential of ensemble learning techniques for complex medical image classification tasks. Future research could incorporate advanced regularization methods, such as MixUp and CutMix, and employ differential learning rates to fine-tune models further. Additionally, exploring transformer-based architectures and attention mechanisms could provide superior feature extraction and contextual understanding compared to traditional CNNs. Expanding datasets to include diverse clinical and geographic scenarios will also enhance model robustness and applicability. Overall, this research highlights the promise of ensemble methods in achieving reliable, generalizable, and accurate solutions for medical image classification, paving the way for more effective diagnostic tools in healthcare.

REFERENCES:

[1] P. Ansari, N. Tabasumma, N. N. Snigdha, N. H. Siam, R. V. N. R. S. Panduru, S. Azam, J. M. A. Hannan, and Y. H. A. Abdel-Wahab, "Diabetic Retinopathy: An Overview on Mechanisms, Pathophysiology and Pharmacotherapy," *Diabetology*, vol. 3, pp. 159–175, 2022. DOI: 10.3390/diabetology3010011.

[2] International Diabetes Federation, "IDF Diabetes Atlas," 8th ed., Brussels, Belgium, 2017.

[3] K. Oh, H. M. Kang, D. Leem, et al., "Early detection of diabetic retinopathy based on deep learning and ultra-wide-field fundus images," *Scientific Reports*, vol. 11, no. 1897, 2021. DOI: 10.1038/s41598-021-81539-3.

[4] G. S. Crabtree and J. S. Chang, "Management of Complications and Vision Loss from Proliferative Diabetic Retinopathy," *Current Diabetes Reports*, vol. 21, no. 33, 2021. DOI: 10.1007/s11892-021-01396-2.

[5] E. Striglia, A. Caccioppo, N. Castellino, M. Reibaldi, and M. Porta, "Emerging drugs for the treatment of diabetic retinopathy," *Expert Opinion on Emerging Drugs*, vol. 25, no. 3, pp. 261–271, 2020. DOI: 10.1080/14728214.2020.1801631.

[6] F. I. Himasa, M. Singhal, A. Ojha, and B. Kumar, "Prospective for Diagnosis and Treatment of Diabetic Retinopathy," *Current Pharmaceutical Design*, vol. 28, no. 7, pp. 560–569, 2022.

[7] G. Mushtaq and F. Siddiqui, "Detection of diabetic retinopathy using deep learning methodology," *IOP Conf. Ser.: Mater. Sci. Eng.*, vol. 1070, p. 012049, 2021.

[8] P. Romero-Aroca, et al., "Validation of a deep learning algorithm for diabetic retinopathy," *Telemedicine and e-Health*, vol. 26, no. 8, pp. 1001–1009, 2020.

[9] A. Esteva, B. Kuprel, R. Novoa, et al., "Dermatologist-level classification of skin cancer with deep neural networks," *Nature*, vol. 542, pp. 115–118, 2017.

[10] I. Y. Abushawish, et al., "Deep learning in automatic diabetic retinopathy detection and grading systems: A comprehensive survey and comparison of methods," *IEEE Access*, vol. 12, pp. 84785–84802, 2024.

[11] J. D. Bodapati and B. B. Balaji, "Self-adaptive stacking ensemble approach with attention based deep neural network models for diabetic retinopathy severity prediction," *Multimedia Tools and Applications*, vol. 83, no. 1, pp. 1083–1102, 2024.

[12] P. Enkvetchakul, O. Surinta, and S. Noppitak, "Effective Data Resampling and Meta-learning Convolutional Neural Networks for Diabetic Retinopathy recognition," *ICIC Express Letters, Part B: Applications*, vol. 13, pp. 939–948, 2022.

[13] J. D. Bodapati, N. S. Shaik, and V. N. Naralasetti, "Composite deep neural network with gated-attention mechanism for diabetic retinopathy severity classification," *Journal of Ambient Intelligence and Humanized Computing*, vol. 12, no. 10, pp. 9825–9839, 2021.

[14] G. T. Reddy, et al., "An ensemble based machine learning model for diabetic retinopathy classification," in *2020 International Conference on Emerging Trends in Information Technology and Engineering (IC-ETITE)*, IEEE, 2020.

[15] M. T. Ribeiro, S. Singh, and C. Guestrin, "Why should I trust you? Explaining the predictions of any classifier," in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, San Francisco, CA, USA, 2016, pp. 1135–1144.

[16] B. Tymchenko, P. Marchenko, and D. Spodarets, "Deep learning approach to diabetic retinopathy detection," *arXiv preprint arXiv:2003.02261*, 2020.

[17] A. Samanta, et al., "Automated detection of diabetic retinopathy using convolutional neural networks on a small dataset," *Pattern Recognition Letters*, vol. 135, pp. 293–298, 2020.

[18] E. M. F. El Houby, "Using transfer learning for diabetic retinopathy stage classification," *Applied Computing and Informatics*, 2021.

[19] P. Zang, et al., "A diabetic retinopathy classification framework based on deep-learning analysis of OCT angiography," *Translational Vision Science & Technology*, vol. 11, no. 7, p. 10, 2022.

[20] J. Arrieta, O. J. Perdomo, and F. A. González, "Deep semi-supervised and self-supervised learning for diabetic retinopathy detection," in *18th International Symposium on Medical Information Processing and Analysis*, vol. 12567, SPIE, 2023.

[21] L. Ma, et al., "Joint ordinal regression and multiclass classification for diabetic retinopathy grading with transformers and CNNs fusion network," *Applied Intelligence*, vol. 53, no. 22, pp. 27505–27518, 2023.

[22] M. Khuntia, P. K. Sahu, and S. Devi, "Novel Strategies Employing Deep Learning Techniques for Classifying Pathological Brain from MR Images," *International Journal of Advanced Computer Science and Applications*, vol. 13, no. 11, 2022.

[23] M. Khuntia, P. K. Sahu, and S. Devi, "Prediction of Presence of Brain Tumor Utilizing Some State-of-the-Art Machine Learning Approaches," *International Journal of Advanced Computer Science and Applications*, vol. 13, no. 5, 2022.

[24] Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the inception architecture for computer vision," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 2818–2826.

[25] F. Chollet, "Xception: Deep learning with depthwise separable convolutions," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 1251–1258.