MID-TERM REPORT

Automated Diabetic Retinopathy Prediction System

BE (CSE) 7th Semester

Computer Science and Engineering

University Institute of Engineering and Technology Panjab University,

Chandigarh – 160014, BHARAT 2024

Minor Project (Jul - Dec 2024)

Submitted By:

SHIVANSHU SAWAN - UEM213119

ZUL QUARNAIN AZAM – UEM213124

KSHITIJ NEGI – UEM213117

Project Guide:

Dr. Savita Gupta

Submitted To:

Dr. Ravreet Kaur

Computer Science Engineering UIET,

Panjab University

ABSTRACT

This report outlines the **progress of our project** on diabetic retinopathy detection using deep learning, specifically convolutional neural networks (CNNs). The project uses a dataset sourced from Kaggle containing retina images, and several preprocessing steps like resizing, labeling, and augmentation have been employed to prepare the data for training the model. The system will be deployed on a Django framework to make the tool accessible for medical professionals. Up to this point, we have completed the dataset preparation, segregation into training, validation, and testing sets, and initiated the model training. The results so far show promising accuracy, and further improvements will be made. Future work will focus on increasing the model's accuracy, exploring other AI algorithms, and integrating additional performance metrics to evaluate the model comprehensively. The final tool will be integrated into a web application using Django.

TABLE OF CONTENT

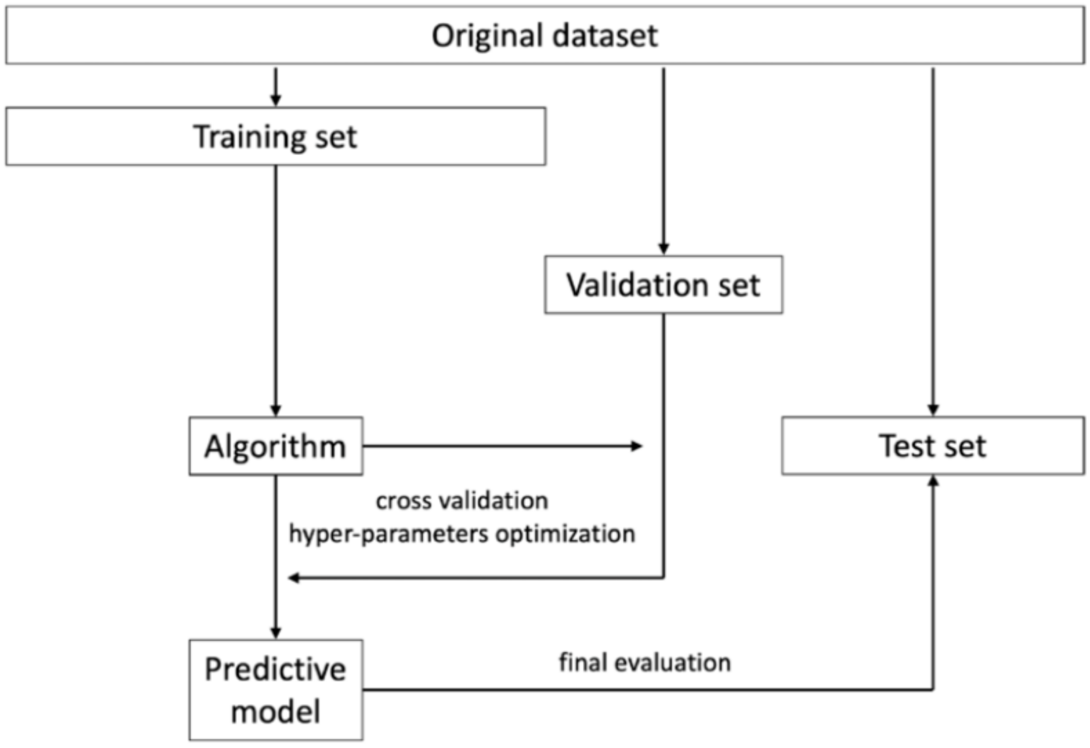
INTRODUCTION & BACKGROUND

Early and accurate detection of diabetic retinopathy, especially during its initial stages, is challenging for medical professionals. Artificial intelligence (AI) and deep learning techniques can assist in identifying early signs of this disease, providing a reference for clinicians to make more informed decisions. In this project, we utilize an open-source retina image dataset from Kaggle.

We preprocess the images through techniques like resizing, normalization, and augmentation to optimize them for model training. We use a Convolutional Neural Network (CNN), which excels in image classification tasks. The dataset is divided into three parts: train, validation and test dataset.

The CNN model's performance is evaluated using precision, recall, and F1 score. The best-performing model will be deployed into a web application using the Django framework to provide automated, real-time diagnostic assistance.

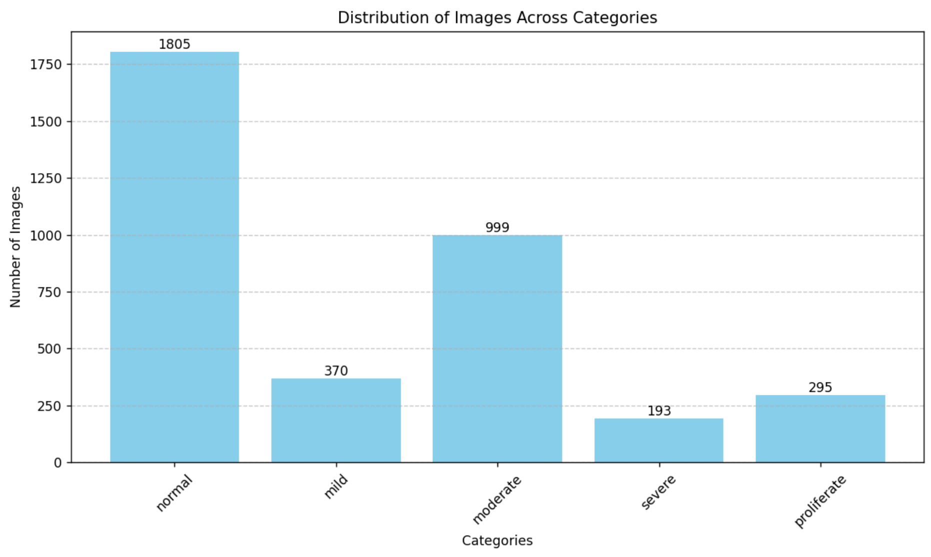
Following is the working sequence of the proposed diabetic retinopathy detection model.

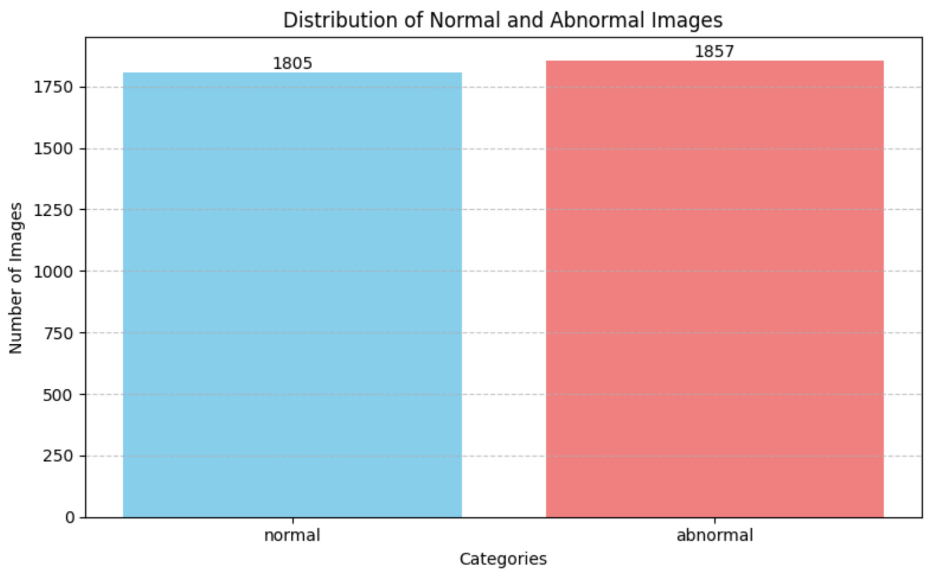


SPECIFICATION AND DESIGN

Dataset

The dataset consists of retina images of various resolutions. Images are resized to a uniform size suitable for the CNN model to process. Labels range from "No DR" to "Proliferative DR," with corresponding severity levels. It contains 3662 files belonging to 5 classes.





Data Preprocessing
Several preprocessing techniques were applied to optimize the dataset for training:

- **Resizing:** All images were resized to a standard resolution to fit the CNN input requirements.
- **Labeling:** Images were labeled based on the severity of diabetic retinopathy.
- **Data Augmentation:** Techniques such as horizontal flips, rotations, and brightness adjustments were used to increase the variability of the dataset and prevent overfitting.
- **Normalization:** Pixel values were normalized to enhance the training performance of the CNN.

AI Algorithm
The primary algorithm used for this project is a Convolutional Neural Network (CNN). The architecture consists of multiple convolutional layers followed by pooling layers, designed to extract features from the input images effectively. The architecture includes:

- **Convolutional Layers:** These layers apply filters to the input images, allowing the model to learn various patterns and features. The initial layers focus on simple features like edges, while deeper layers capture more complex patterns.
- **Pooling Layers:** Max pooling layers are used to down-sample the feature maps, reducing dimensionality and computational complexity while retaining essential information.
- **Fully Connected Layers:** After the convolutional and pooling layers, the output is flattened and passed through one or more fully connected layers, which contribute to the final classification decision.

Deployment
The model will be deployed on the Django framework, which will serve as a backend for integrating the trained model into a web-based tool accessible by medical professionals.

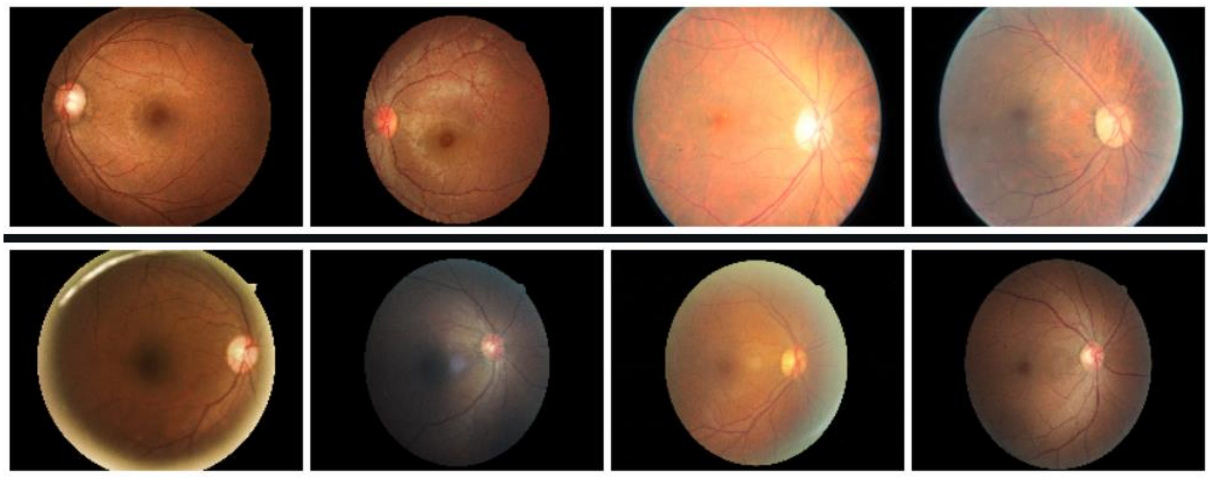
PROGRESS TILL DATE ALONG WITH RESULTS

Dataset Download

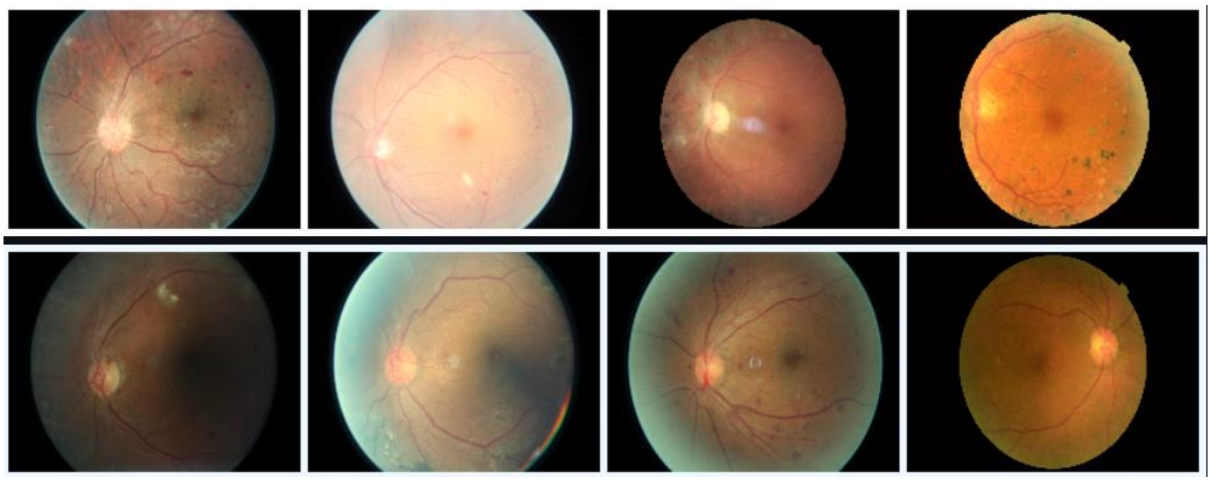
The retina image dataset, sourced from Kaggle's "**Diabetic Retinopathy (2019 Data)**", has been downloaded. The dataset contains high-resolution retina images categorized based on five classes of diabetic retinopathy severity: No DR, Mild, Moderate, Severe, and Proliferative DR.

Kaggle dataset link: https://www.kaggle.com/datasets/sovitrath/diabetic-retinopathy-224x224-2019-data

No Retinopathy



Proliferative Retinopathy

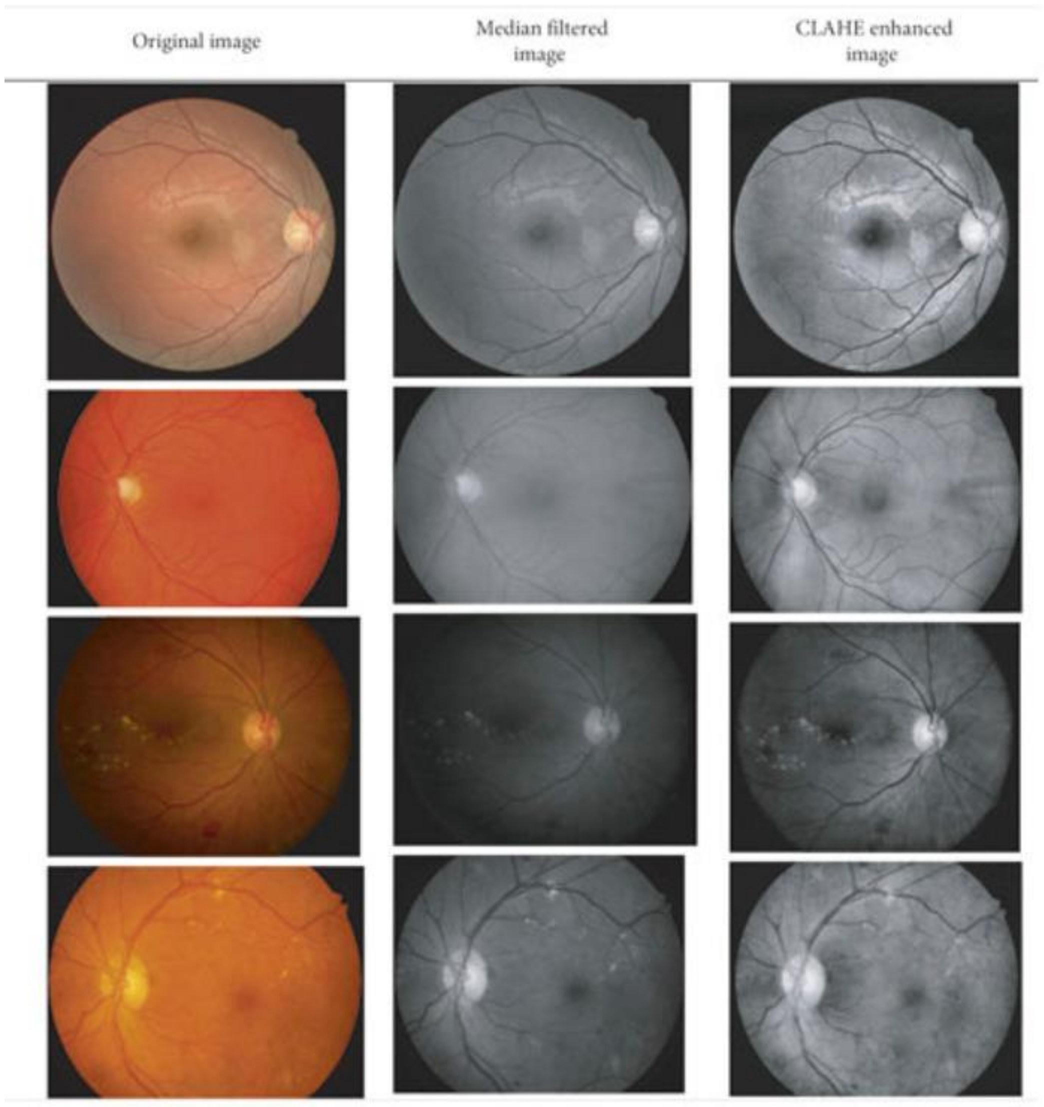


Data Preprocessing

Significant preprocessing has been completed to prepare the dataset for training:

The preprocessing pipeline is the following:

- Resize images to 256x256 resolution
- Remove totally black images form dataset
- Rotate and mirror(Rotate DR images to 90º,120º,180º,270º + mirror, and only mirror non-DR images)
- Convert image to grayscale image
- Image Denoising using Median Filter
- CLAHE (Contrast Limited Adaptive Histogram Equalization)



Data Segregation

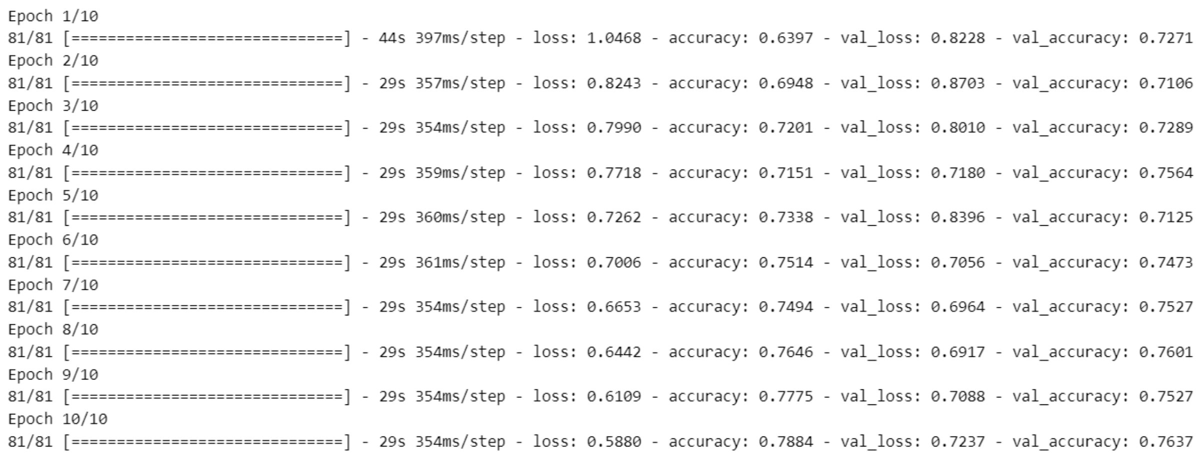
The dataset has been split into three subsets to evaluate the model's performance:

- **Training Set:** 70% of the data has been allocated for training.
- **Validation Set:** 15% of the data is used for model validation to fine-tune hyperparameters and avoid overfitting.
- **Test Set:** The remaining 15% is reserved for testing the final model's performance on unseen data.

Training the CNN Model

We are using a CNN architecture designed to handle the complexity of retina images. The architecture includes:

- Several convolutional layers with ReLU activations to capture hierarchical spatial features.
- MaxPooling layers to reduce dimensionality and computational cost.
- Dense layers leading to a softmax output layer for classification.

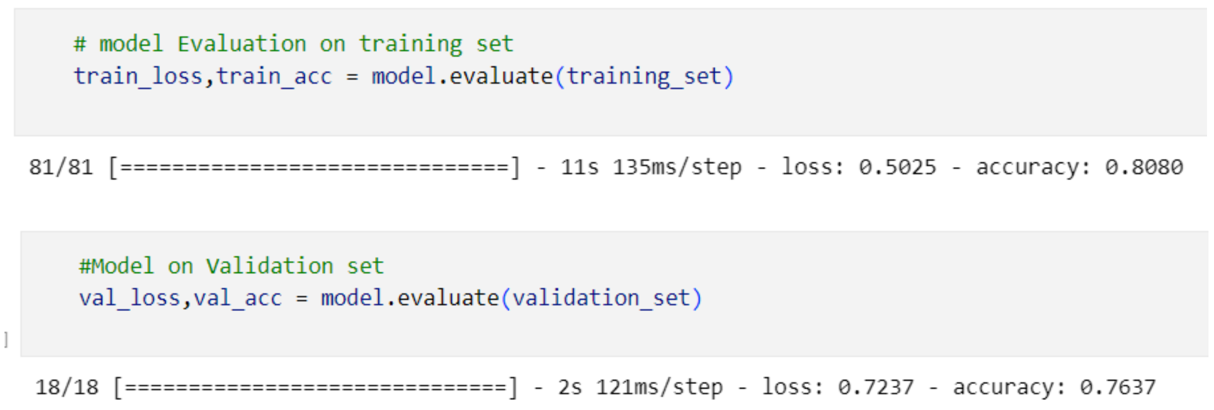


```
Epoch 1/10
81/81 [==============================] - 44s 397ms/step - loss: 1.0468 - accuracy: 0.6397 - val_loss: 0.8228 - val_accuracy: 0.7271
Epoch 2/10
81/81 [==============================] - 29s 357ms/step - loss: 0.8243 - accuracy: 0.6948 - val_loss: 0.8703 - val_accuracy: 0.7106
Epoch 3/10
81/81 [==============================] - 29s 354ms/step - loss: 0.7990 - accuracy: 0.7201 - val_loss: 0.8010 - val_accuracy: 0.7289
Epoch 4/10
81/81 [==============================] - 29s 359ms/step - loss: 0.7718 - accuracy: 0.7151 - val_loss: 0.7180 - val_accuracy: 0.7564
Epoch 5/10
81/81 [==============================] - 29s 360ms/step - loss: 0.7262 - accuracy: 0.7338 - val_loss: 0.8396 - val_accuracy: 0.7125
Epoch 6/10
81/81 [==============================] - 29s 361ms/step - loss: 0.7006 - accuracy: 0.7514 - val_loss: 0.7056 - val_accuracy: 0.7473
Epoch 7/10
81/81 [==============================] - 29s 354ms/step - loss: 0.6653 - accuracy: 0.7494 - val_loss: 0.6964 - val_accuracy: 0.7527
Epoch 8/10
81/81 [==============================] - 29s 354ms/step - loss: 0.6442 - accuracy: 0.7646 - val_loss: 0.6917 - val_accuracy: 0.7601
Epoch 9/10
81/81 [==============================] - 29s 354ms/step - loss: 0.6109 - accuracy: 0.7775 - val_loss: 0.7088 - val_accuracy: 0.7527
Epoch 10/10
81/81 [==============================] - 29s 354ms/step - loss: 0.5880 - accuracy: 0.7884 - val_loss: 0.7237 - val_accuracy: 0.7637
```

Training Progress

So far, the model has been trained on multiple epochs. Early results show a steady improvement in accuracy across the training and validation datasets:

- **Initial Accuracy:** Starting with a modest accuracy of around **63%,** after hyperparameter tuning (learning rate, batch size, etc.), the current training accuracy is at **80%**.
- **Validation Accuracy:** The validation accuracy stands at approximately **76%,** indicating a well-fitted model with minimal overfitting.



```
# model Evaluation on training set
train_loss,train_acc = model.evaluate(training_set)
```

```
81/81 [==============================] - 11s 135ms/step - loss: 0.5025 - accuracy: 0.8080
```

```
#Model on Validation set
val_loss,val_acc = model.evaluate(validation_set)
```

```
18/18 [==============================] - 2s 121ms/step - loss: 0.7237 - accuracy: 0.7637
```

FUTURE WORK PLAN

- **Improving the Accuracy:** Several techniques will be explored to improve model accuracy, including hyperparameter tuning, increasing the depth of the CNN, and experimenting with other model architectures such as transfer learning with pre-trained models like ResNet or Inception.
- **Research on Other AI Algorithms:** Further research will be conducted to explore other deep learning models and algorithms to enhance model performance.
- **Other Metrics for Model Evaluation:** In addition to accuracy, metrics such as F1-score, precision, recall, and AUC-ROC curves will be used to evaluate the model's performance comprehensively.
- **Integration into Web Tool with Django:** The final model will be integrated into a web tool using the Django framework, allowing users to upload retina images and receive real-time diabetic retinopathy severity predictions.