Index

Acknowledgement

I would like to express my sincere gratitude to all those who contributed to the successful completion of this project.

First and foremost, I am thankful to the **Indian Diabetic Retinopathy Image Dataset (IDRiD)** providers for making high-quality medical imaging data publicly available. Their contributions have enabled researchers and developers to work towards improving healthcare solutions.

I extend my appreciation to the **open-source communities**, including the developers and maintainers of libraries such as **OpenCV, Scikit-learn, NumPy, Pandas, and Matplotlib**. Their continuous efforts in developing and maintaining these tools have greatly facilitated this project.

I am also grateful to my **mentors, professors, and peers**, whose guidance and valuable insights helped shape the direction of this work. Their feedback and support have been instrumental in improving the project's methodology and outcomes.

Lastly, I would like to acknowledge the **researchers and medical professionals** whose extensive work in the field of diabetic retinopathy and medical image analysis has inspired this project. Their research has provided the foundation for developing AI-driven diagnostic tools that contribute to the early detection of diabetic retinopathy.

Project Overview

Diabetic Retinopathy (DR) is a severe complication of diabetes that can lead to vision impairment or blindness if not detected and treated early. With the increasing prevalence of diabetes worldwide, early detection of DR has become crucial for timely medical intervention.

This project focuses on the **automatic classification of Diabetic Retinopathy** using **machine learning techniques**. It leverages the **Indian Diabetic Retinopathy Image Dataset (IDRiD)**, which consists of retinal fundus images labeled according to DR severity.

A **Random Forest Classifier** is trained on extracted image features to classify fundus images into **normal or diabetic retinopathy cases**. The process involves:

1. **Image Preprocessing:** Resizing images, extracting the green channel, and applying **Contrast Limited Adaptive Histogram Equalization (CLAHE)** for enhancement.
2. **Feature Extraction:** Using **Gray-Level Co-occurrence Matrix (GLCM)** to extract key textural features.
3. **Model Training & Classification:** Using **Random Forest**, a powerful ensemble learning technique, for classification.
4. **Evaluation Metrics:** Measuring **accuracy, sensitivity, specificity, F1-score**, and visualizing results through **scatter plots, line graphs, histograms, and bar charts**.

This automated classification system can assist **ophthalmologists** by providing a preliminary diagnosis, reducing manual effort, and speeding up the screening process for diabetic retinopathy detection.

Random Forest Classifier

Random Forest is an ensemble machine learning algorithm that combines multiple decision trees to improve prediction accuracy and reduce overfitting. It is widely used for classification and regression tasks.

How Random Forest Works?

1. **Multiple Decision Trees** → Random Forest builds several decision trees from different subsets of data.
2. **Feature Randomness** → Each tree is trained on a random subset of features, ensuring diversity in learning.
3. **Majority Voting** → For classification, each tree votes for a class, and the majority class is chosen as the final prediction.

Why Use Random Forest?

- **Handles High-Dimensional Data** → Works well with multiple features, such as GLCM texture features.
- **Reduces Overfitting** → Unlike single decision trees, Random Forest generalizes well.
- **Handles Missing Data & Noisy Data** → Robust in real-world scenarios.
- **High Accuracy & Stability** → Performs better than individual decision trees.

Random Forest in This Project

- Used **100 decision trees** to classify retinal images as **Normal or Diabetic Retinopathy-affected**.
- Trained on **GLCM texture features** extracted from preprocessed retinal images.
- Achieved an **accuracy of 62.14%**, with **high sensitivity (86.96%)** but **low specificity (11.76%)**.
- Suitable for this project due to its ability to work well with **structured numerical features**.

Dataset Description

The project utilizes the **Indian Diabetic Retinopathy Image Dataset (IDRiD)**, which contains **retinal fundus images** labeled for **Diabetic Retinopathy (DR) severity levels**. This dataset is structured into multiple folders for images and corresponding ground truth labels.

1. Dataset Used

- **Indian Diabetic Retinopathy Image Dataset (IDRiD)**
- Provides **high-resolution fundus images** with labeled information.
- Includes **annotations for DR severity** based on medical diagnosis.

2. Folder Structure

📁 1. Original Images/

- 📂 **a. Training Set/** → Contains images used for training the model.
- 📂 **b. Testing Set/** → Contains images for testing and evaluation.

📁 2. Groundtruths/

- 📄 **a. IDRiD_Disease Grading_Training Labels.csv** → Labels for training images.
- 📄 **b. IDRiD_Disease Grading_Testing Labels.csv** → Labels for testing images.

3. Label Information

The dataset’s CSV files contain essential information about each image:

Column Name	**Description**
Image name	Corresponds to the retinal image filename.
Retinopathy grade	Severity level of DR (0 = Normal, 1-4 = DR stages).
Risk of macular edema	Risk level of macular edema (Not used in this classification task).

4. Data Distribution

- The dataset contains a mix of **normal and diabetic** fundus images.
- A balanced dataset is crucial for **reducing bias** in classification.
- **Preprocessing techniques** (such as **data augmentation**) can help improve model performance.

Project Workflow

1. Import Required Libraries

- Load essential **Python libraries** such as **OpenCV, NumPy, Pandas, Matplotlib**, and **Scikit-learn** for data handling, preprocessing, model training, and evaluation.

2. Load Dataset & Labels

- Read **IDRiD dataset CSV files** containing image labels (retinopathy grade).
- Convert the **Retinopathy grade** column into **binary classification**:
  - **0 = Normal (No DR)**
  - **1 = Diabetic Retinopathy (Any DR Stage)**

3. Image Preprocessing

- Resize images to a **fixed resolution (256x256 pixels)**.
- Extract the **green channel** (useful for detecting abnormalities).
- Apply **Contrast Limited Adaptive Histogram Equalization (CLAHE)** to enhance image details.

4. Feature Extraction using GLCM (Gray-Level Co-occurrence Matrix)

- Compute **texture-based features** from the preprocessed images, including:
    - **Contrast**
    - **Correlation**
    - **Energy**
    - **Homogeneity**
- These features help the model **distinguish between normal and diabetic retinopathy images**.

5. Train Random Forest Classifier

- Split data into **training (80%)** and **testing (20%)** sets.
- Train a **Random Forest model** with 100 estimators.
- Fit extracted features to the classifier.

6. Model Evaluation

- Predict results on the **test dataset**.
- Compute key performance metrics:
    - **Accuracy Score**
    - **Confusion Matrix**
    - **Classification Report** (Precision, Recall, F1-score)

7. Generate Visualizations

- **Scatter Plot**: Visualizes feature distribution.

- **Line Plot**: Compares actual vs predicted values.
- **Histogram**: Shows feature distribution.
- **Bar Chart**: Displays class distribution.

8. Results & Performance Analysis

- Analyze **model accuracy and errors** using the confusion matrix.
- Identify **misclassifications** and possible improvements.
- Compare results with **alternative approaches** like **Deep Learning (CNNs)**.

9. Future Enhancements

- Implement **CNN-based deep learning models** for improved feature extraction.
- Apply **data augmentation** to increase dataset size and balance classes.
- Explore **other ML algorithms** like **SVM, XGBoost** for comparison.

Step-by-Step Implementation

Step 1: Import Required Libraries

```
import os
import cv2
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from skimage.feature import graycomatrix, graycoprops
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, confusion_matrix,
classification_report
```

Libraries Used:

- **OpenCV** → Image processing
- **NumPy, Pandas** → Data handling
- **Matplotlib, Seaborn** → Visualization
- **Scikit-learn** → ML model & evaluation
- **Skimage** → Feature extraction using GLCM

Step 2: Define Paths

```
DATASET_PATH = r"D:\bvp\SEM-6\DV\project\IDRiD_Dataset\B. Disease Grading\1.
Original Images"
LABELS_TRAIN_PATH = r"D:\bvp\SEM-6\DV\project\IDRiD_Dataset\B. Disease
Grading\2. Groundtruths\a. IDRiD_Disease Grading_Training Labels.csv"
```

```
LABELS_TEST_PATH    =    r"D:\bvp\SEM-6\DV\project\IDRiD_Dataset\B.    Disease
Grading\2. Groundtruths\b. IDRiD_Disease Grading_Testing Labels.csv"
```

Step 3: Load Dataset & Labels

```
def load_labels(labels_path):
    labels_df = pd.read_csv(labels_path)
    labels_df['Binary_Class'] = labels_df['Retinopathy grade'].apply(lambda x:
0 if x == 0 else 1)  # Normal = 0, DR = 1
    return labels_df

labels_train = load_labels(LABELS_TRAIN_PATH)
labels_test = load_labels(LABELS_TEST_PATH)
```

Key Operations:

- Reads CSV file containing labels.
- Converts retinopathy grade into binary classification (0: Normal, 1: Diabetic Retinopathy).

Step 4: Feature Extraction using GLCM

```
def extract_features(image):
    gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
    glcm   =   graycomatrix(gray,   distances=[1],   angles=[0],   levels=256,
symmetric=True, normed=True)

    contrast = graycoprops(glcm, 'contrast')[0, 0]
    dissimilarity = graycoprops(glcm, 'dissimilarity')[0, 0]
    homogeneity = graycoprops(glcm, 'homogeneity')[0, 0]
    energy = graycoprops(glcm, 'energy')[0, 0]
    correlation = graycoprops(glcm, 'correlation')[0, 0]

    return [contrast, dissimilarity, homogeneity, energy, correlation]
```

Features Computed:

- **Contrast** → Measures intensity variations.
- **Dissimilarity** → Measures variations in neighboring pixels.
- **Homogeneity** → Measures smoothness.
- **Energy** → Measures uniformity.
- **Correlation** → Measures pixel dependency.

Step 5: Image Preprocessing

```
def preprocess_image(img_path):
    image = cv2.imread(img_path)
    image = cv2.resize(image, (256, 256))  # Resize image

    green_channel = image[:, :, 1]  # Extract Green Channel

    # Apply CLAHE (Contrast Enhancement)
    clahe = cv2.createCLAHE(clipLimit=2.0, tileGridSize=(8, 8))
    enhanced_image = clahe.apply(green_channel)

    # Convert back to 3-channel image for feature extraction
    enhanced_image = cv2.merge([enhanced_image, enhanced_image,
enhanced_image])

    # Extract Features
    features = extract_features(enhanced_image)

    return enhanced_image, features
```

Key Operations:

- Resizes images to **256×256**.

- Extracts the **green channel** for better contrast.
- Applies **CLAHE** for enhanced visibility of retinal structures.

Step 6: Load Images & Extract Features

```
def load_dataset(images_folder, labels_df):
    X = []
    y = []

    for _, row in labels_df.iterrows():
        img_name = row['Image name'] + ".jpg"
        img_path = os.path.join(images_folder, img_name)

        if os.path.exists(img_path):
            _, features = preprocess_image(img_path)
            X.append(features)
            y.append(row['Binary_Class'])
        else:
            print(f"Image not found: {img_path}")

    return np.array(X), np.array(y)
```

Key Operations:

- Loops through images to process and extract features.
- Stores extracted **features and labels**.

```
X_train, y_train = load_dataset(os.path.join(DATASET_PATH, "a. Training Set"),
labels_train)
X_test, y_test = load_dataset(os.path.join(DATASET_PATH, "b. Testing Set"),
labels_test)
```

Step 7: Train Random Forest Classifier

```
rf_model = RandomForestClassifier(n_estimators=100, random_state=42)
rf_model.fit(X_train, y_train)
```

Key Operations:

- Trains **Random Forest** model with 100 trees.
- Uses extracted **GLCM features** for training.

Step 8: Model Evaluation

```
y_pred = rf_model.predict(X_test)

accuracy = accuracy_score(y_test, y_pred)
conf_matrix = confusion_matrix(y_test, y_pred)
report = classification_report(y_test, y_pred, target_names=["Normal",
"Diabetic Retinopathy"])
```

Key Metrics:

- **Accuracy**
- **Confusion Matrix**
- **Classification Report**

```
sns.heatmap(conf_matrix, annot=True, fmt="d", cmap="Blues",
xticklabels=["Normal", "Diabetic"], yticklabels=["Normal", "Diabetic"])
plt.xlabel("Predicted")
plt.ylabel("Actual")
plt.title("Confusion Matrix")
plt.show()
```

Step 9: Generate Visualizations

Scatter Plot

```
plt.scatter(X_train[:, 0], X_train[:, 1], c=y_train, cmap="coolwarm")
plt.title("Scatter Plot of Feature 1 vs Feature 2")
plt.xlabel("Feature 1")
plt.ylabel("Feature 2")
plt.grid()
plt.show()
```

Line Plot

```
plt.plot(sorted(y_test), label="Actual")
plt.plot(sorted(y_pred), label="Predicted", linestyle="dashed")
plt.title("Line Plot of Predictions")
plt.legend()
plt.grid()
plt.show()
```

Histogram

```
plt.hist(X_train[:, 0], bins=20, color="skyblue", edgecolor="black")
plt.title("Histogram of Feature 1")
plt.xlabel("Feature 1")
plt.ylabel("Frequency")
plt.grid()
plt.show()
```

Bar Chart

```
labels = ["Normal", "Diabetic"]
values = [sum(y_train == 0), sum(y_train == 1)]
plt.bar(labels, values, color=["green", "red"])
plt.title("Bar Chart of Class Distribution")
plt.ylabel("Count")
plt.grid()
plt.show()
```

Results & Performance Metrics

1. Model Accuracy

- **Accuracy: 62.14%**
- While the model performs moderately well, improvements can be made by tuning hyperparameters or adding more training data.

2. Confusion Matrix Analysis

- The model correctly identifies **Diabetic Retinopathy cases (87% Recall)** but struggles with Normal cases (**12% Recall**).
- **Specificity is low (11.76%)**, meaning many Normal cases are misclassified.

3. Classification Report Insights

Class	Precision	Recall (Sensitivity)	F1-Score	Support
Normal	0.31	0.12	0.17	34
Diabetic Retinopathy	0.67	0.87	0.75	69

- **Macro Avg F1-Score: 0.46**
- **Weighted Avg F1-Score: 0.56**

🔹 **Diabetic Retinopathy detection is strong**, but Normal cases require improvement.

4. Sensitivity, Specificity & F1-Score

- **Sensitivity (Recall): 86.96%** (Good for detecting Diabetic Retinopathy)
- **Specificity: 11.76%** (Poor at distinguishing Normal cases)
- **F1 Score: 20.73%** (Indicates an imbalance in class predictions)

5. Observations & Recommendations

☑ Strengths:

- High recall for **Diabetic Retinopathy** (87%) → Good at catching positive cases.
- F1-score for **Diabetic Retinopathy (0.75)** suggests a balanced performance for this class.

🚩 Weaknesses:

- **Low specificity (11.76%)** → The model misclassifies many Normal cases as Diabetic Retinopathy.
- **Class imbalance issue** → Needs better handling of Normal case predictions.

🔧 Possible Improvements:

- **Data Balancing:** Use **SMOTE** or **class weighting** to enhance Normal case detection.
- **Feature Engineering:** Add additional features to improve classification.
- **Model Tuning:** Experiment with **hyperparameter tuning** to optimize performance.

Conclusion

The developed **Diabetic Retinopathy classification model** demonstrates **moderate accuracy (62.14%)**, performing well in detecting **Diabetic Retinopathy cases (87% recall)**. However, it struggles with distinguishing **Normal cases (12% recall, 11.76% specificity)**, leading to misclassification.

Key Takeaways:

☑ Strengths:

- High recall for **Diabetic Retinopathy** detection → Effectively identifies affected cases.
- Acceptable **F1-score (0.75) for Diabetic Retinopathy** → Indicates reliable classification for this class.

🚩 Limitations:

- **Low specificity (11.76%)** → Many Normal cases are misclassified.
- **Class imbalance affects performance** → Normal cases are not well recognized.

Future Improvements

1. Data Balancing & Augmentation

- **Use SMOTE (Synthetic Minority Over-sampling Technique)** to handle class imbalance.
- **Augment images** (rotation, flipping, brightness adjustments) to improve model generalization.

2. Advanced Feature Extraction

- Experiment with **additional texture features** (e.g., entropy, variance) to improve discrimination.
- **Use deep learning-based feature extraction** with pre-trained CNN models (e.g., ResNet, VGG16).

3. Model Optimization

- Fine-tune **hyperparameters** of the Random Forest classifier to reduce overfitting.
- Explore **alternative ML models** (e.g., SVM, XGBoost) for better performance.

4. Deep Learning Approaches

- Implement **CNN-based models** for automatic feature extraction.
- Try **transfer learning** to leverage pre-trained networks for improved accuracy.

Final Verdict

The implemented **Random Forest model** successfully classifies **Diabetic Retinopathy** using **GLCM-based texture features** with an **accuracy of 62.14%**. While the model demonstrates **high sensitivity (86.96%)**, it struggles with **low specificity (11.76%)**, indicating a tendency to misclassify normal cases.

Key Takeaways:

✅ **Effective in detecting DR** but needs improvements in distinguishing normal cases.
✅ **GLCM features** provide valuable texture-based insights but might not be sufficient alone.
✅ **ML approach shows promise**, but **deep learning-based feature extraction** could significantly enhance performance.

While this approach provides a **solid baseline**, further optimizations in **feature selection, data balancing, and model choice** can lead to a more **robust and reliable** DR classification system.

References

- **IDRiD Dataset – Diabetic Retinopathy Grading (IEEE)**

🔗 https://ieee-dataport.org/open-access/indian-diabetic-retinopathy-image-dataset-idrid

- **Scikit-Image Documentation** – GLCM Feature Extraction

🔗 https://scikit-image.org/docs/stable/

- **Scikit-Learn Documentation** – Random Forest Classifier

🔗 https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html

- **OpenCV Documentation** – Image Processing

🔗 https://docs.opencv.org/

- **Matplotlib & Seaborn Documentation** – Data Visualization

🔗 https://matplotlib.org/stable/

🔗 https://seaborn.pydata.org/

- **Pandas Documentation** – Data Handling

🔗 https://pandas.pydata.org/docs/