1. Project 2: Diabetic Retinopathy Classification

The following findings and results are based on the machine learning analysis of the provided dataset i.e. Diabetic Retinopathy (DR) data. The applied deep learning approach demonstrates the application of 3 different deep learning models *ResNet-50, VGG-16* and *Inception-v3* along with one machine learning classification methods/model *Random Forest*.

i. Dataset Split

The provided data was split into 80% for training and 20% on testing. The results are based on the test subset. The following highlights the implementation of the split during the

```python
# Set up directories for images
image_dir = 'Desktop/My Courses/Spring 2025/Machine Learning/Final Project/retinopathy_3Classes'  # path to images
labels = []  # List to hold labels (0, 1, 2 for healthy, moderate DR, severe DR)
image_paths = []  # List to hold image paths

# Collect image paths and corresponding labels
for filename in os.listdir(image_dir):
    if filename.endswith('.jpeg'):
        id_side_class = filename.split('_')  # ID_Side_Class.jpeg
        label = int(id_side_class[-1].split('.')[0])  # Last part is the class
        image_paths.append(os.path.join(image_dir, filename))
        labels.append(label)

# Convert labels to numpy array
labels = np.array(labels)

# Load and preprocess images
images = []
for path in image_paths:
    img = tf.keras.preprocessing.image.load_img(path, target_size=(224, 224))
    img_array = tf.keras.preprocessing.image.img_to_array(img) / 255.0  # Normalize
    images.append(img_array)

images = np.array(images)

# Split into training and testing sets (80% / 20%)
X_train, X_test, y_train, y_test = train_test_split(images, labels, test_size=0.2, stratify=labels, random_state=42)
```

ii. Three Deep Learning and One ML Classification Methods

For this purpose, three DL/CNN and one ML models were used. Images were analyzed with DL models like *ResNet-50, VGG-16* and *Inception-v2* for DL/CNN and *Random Forest* for ML.

i. ResNet-50 Model

```python
# Load ResNet50 model
base_model_resnet = ResNet50(weights='imagenet', include_top=False, input_shape=(224, 224, 3))

# Freeze the base model layers
base_model_resnet.trainable = False

# Create the model
model_resnet = models.Sequential([
    base_model_resnet,  # Pre-trained ResNet50 base
    layers.GlobalAveragePooling2D(),  # Global average pooling to reduce the output to a vector
    layers.Dense(3, activation='softmax')  # Output layer with 3 classes
])

# Compile the model
model_resnet.compile(optimizer='adam', loss='sparse_categorical_crossentropy', metrics=['accuracy'])
```

Code for RestNet-50

ii. VGG-16 Model

```
# Load VGG-16 model
base_model_vgg = VGG16(weights='imagenet', include_top=False, input_shape=(224, 224, 3))

# Freeze the base model layers
base_model_vgg.trainable = False

# Create the model
model_vgg = models.Sequential([
    base_model_vgg,  # Pre-trained VGG16 base
    layers.GlobalAveragePooling2D(),  # Global average pooling to reduce the output to a vector
    layers.Dense(3, activation='softmax')  # Output layer with 3 classes
])

# Compile the model
model_vgg.compile(optimizer='adam', loss='sparse_categorical_crossentropy', metrics=['accuracy'])
```

Code for VGG-16 Model

iii. InceptionV3 Model

```
# Load InceptionV3 model
base_model_inception = InceptionV3(weights='imagenet', include_top=False, input_shape=(224, 224, 3))

# Freeze the base model layers
base_model_inception.trainable = False

# Create the model
model_inception = models.Sequential([
    base_model_inception,  # Pre-trained InceptionV3 base
    layers.GlobalAveragePooling2D(),  # Global average pooling to reduce the output to a vector
    layers.Dense(3, activation='softmax')  # Output layer with 3 classes
])

# Compile the model
model_inception.compile(optimizer='adam', loss='sparse_categorical_crossentropy', metrics=['accuracy'])
```

Code for Inception-v3 Model.

4. ML Classification Model

```
# Flatten the images so they can be used in traditional ML models (Random Forest)
X_train_flattened = X_train.reshape(X_train.shape[0], -1)  # Flatten the images into 1D vectors
X_test_flattened = X_test.reshape(X_test.shape[0], -1)

# Train a Random Forest model
rf_model = RandomForestClassifier(n_estimators=100, random_state=42)
rf_model.fit(X_train_flattened, y_train)

# Predictions on the test set
rf_pred = rf_model.predict(X_test_flattened)

# Calculate Accuracy and F1-score
rf_accuracy = accuracy_score(y_test, rf_pred)
rf_f1 = f1_score(y_test, rf_pred, average='weighted')

print(f"Random Forest Accuracy: {rf_accuracy:.4f}")
print(f"Random Forest F1 Score: {rf_f1:.4f}")
```

```
Random Forest Accuracy: 0.5837
Random Forest F1 Score: 0.4876
```

Code for Random Forest Model.

The models were then trained on 10 epochs as a conservative approach to save time and computational resource. 10 epochs is a good chunk as it trains the data just enough to give meaningful results without any risk of overfitting or underfitting.

5. Training Models

```
# Train the ResNet-50 model
history_resnet = model_resnet.fit(X_train, y_train, validation_data=(X_test, y_test), epochs=10, batch_size=32)

# Train the VGG-16 model
history_vgg = model_vgg.fit(X_train, y_train, validation_data=(X_test, y_test), epochs=10, batch_size=32)

# Train the InceptionV3 model
history_inception = model_inception.fit(X_train, y_train, validation_data=(X_test, y_test), epochs=10, batch_size=32)
```

```
Epoch 1/10
53/53 ━━━━━━━━━━━━━━━━━━━━ 50s 900ms/step - accuracy: 0.5361 - loss: 0.9871 - val_accuracy: 0.5837 - val_loss: 0.9479
Epoch 2/10
53/53 ━━━━━━━━━━━━━━━━━━━━ 44s 836ms/step - accuracy: 0.5895 - loss: 0.9496 - val_accuracy: 0.5837 - val_loss: 0.9529
Epoch 3/10
53/53 ━━━━━━━━━━━━━━━━━━━━ 44s 839ms/step - accuracy: 0.5972 - loss: 0.9525 - val_accuracy: 0.5837 - val_loss: 0.9503
```

Code snippet for Model training.

iii. Tables and Figures of Classification, F1-Scores, Accuracies

```
# Get Accuracy and F1-Score for each model
resnet_acc = history_resnet.history['val_accuracy'][-1]
vgg_acc = history_vgg.history['val_accuracy'][-1]
inception_acc = history_inception.history['val_accuracy'][-1]

# Random Forest metrics computed above
rf_acc = rf_accuracy
rf_f1 = rf_f1

# Create a summary table with F1-scores computed
results = {
    'Model': ['ResNet-50', 'VGG-16', 'InceptionV3', 'Random Forest'],
    'Accuracy': [resnet_acc, vgg_acc, inception_acc, rf_acc],
    'F1 Score': [
        f1_score(y_test, model_resnet.predict(X_test).argmax(axis=1), average='weighted'),
        f1_score(y_test, model_vgg.predict(X_test).argmax(axis=1), average='weighted'),
        f1_score(y_test, model_inception.predict(X_test).argmax(axis=1), average='weighted'),
        rf_f1
    ]
}
# Convert to DataFrame for easy viewing
results_df = pd.DataFrame(results)
print(results_df)
```

```
14/14 ━━━━━━━━━━━━━━━━━━━━ 11s 730ms/step
14/14 ━━━━━━━━━━━━━━━━━━━━ 32s 2s/step
14/14 ━━━━━━━━━━━━━━━━━━━━ 10s 639ms/step
           Model  Accuracy  F1 Score
0      ResNet-50  0.583732  0.430304
1         VGG-16  0.586124  0.436243
2    InceptionV3  0.645933  0.613259
3  Random Forest  0.583732  0.487595
```

Model	Accuracy	F1 Score
ResNet-50	**0.583732**	**0.430304**
VGG-16	**0.586124**	**0.436243**
InceptionV3	**0.645933**	**0.613259**
Random Forest	**0.583732**	**0.48759**

Table 1: Accuracy and F1 Score of DL and ML model.

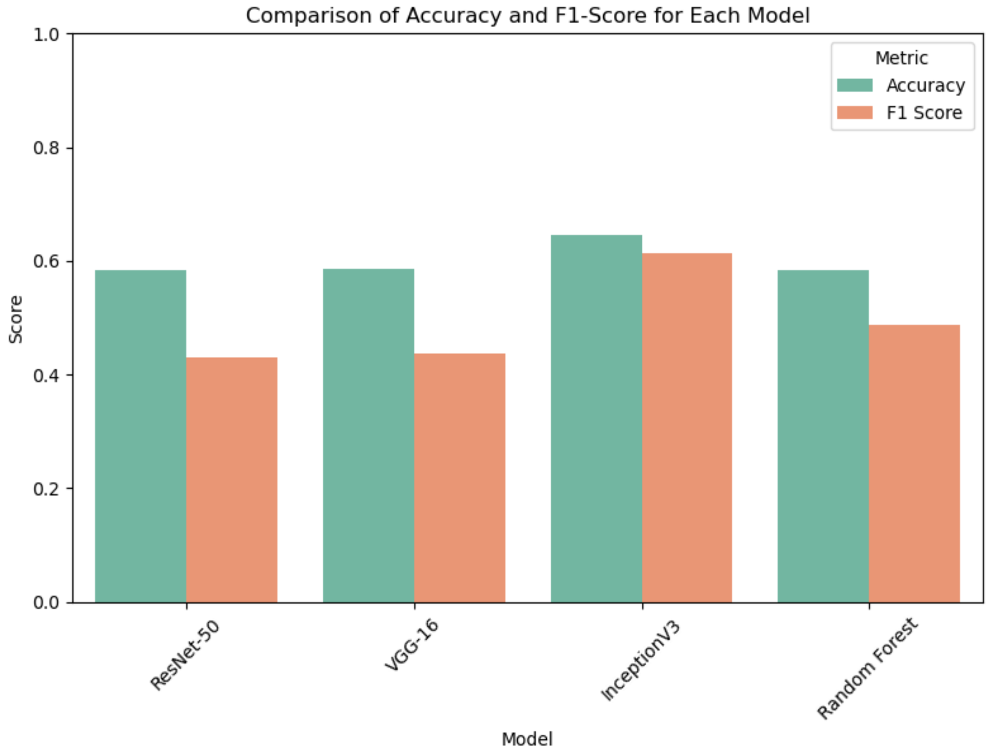

Figure 1: Visualization of Accuracy and F1 Scores of all Models.

iv. Classification Report

```
Classification Report for ResNet-50:
14/14 ━━━━━━━━━━━━━━━━━━━━ 9s 630ms/step
              precision    recall  f1-score   support

           0       0.58      1.00      0.74       244
           1       0.00      0.00      0.00       112
           2       0.00      0.00      0.00        62

    accuracy                           0.58       418
   macro avg       0.19      0.33      0.25       418
weighted avg       0.34      0.58      0.43       418

Classification Report for VGG-16:
14/14 ━━━━━━━━━━━━━━━━━━━━ 33s 2s/step
              precision    recall  f1-score   support

           0       0.59      1.00      0.74       244
           1       0.00      0.00      0.00       112
           2       0.50      0.02      0.03        62

    accuracy                           0.59       418
   macro avg       0.36      0.34      0.26       418
weighted avg       0.42      0.59      0.44       418

Classification Report for InceptionV3:
14/14 ━━━━━━━━━━━━━━━━━━━━ 7s 466ms/step
              precision    recall  f1-score   support

           0       0.70      0.86      0.77       244
           1       0.44      0.21      0.29       112
           2       0.57      0.60      0.58        62

    accuracy                           0.65       418
   macro avg       0.57      0.56      0.55       418
weighted avg       0.61      0.65      0.61       418

Classification Report for Random Forest:
              precision    recall  f1-score   support

           0       0.61      0.94      0.74       244
           1       0.33      0.12      0.17       112
           2       0.40      0.03      0.06        62

    accuracy                           0.58       418
   macro avg       0.45      0.36      0.32       418
weighted avg       0.51      0.58      0.49       418
```

```python
# Classification Reports for each model
print("Classification Report for ResNet-50:") #ResNet-50
print(classification_report(y_test, model_resnet.predict(X_test).argmax(axis=1), zero_division=0))

print("Classification Report for VGG-16:") #VGG-16
print(classification_report(y_test, model_vgg.predict(X_test).argmax(axis=1), zero_division=0))

print("Classification Report for InceptionV3:") #InceptionV3
print(classification_report(y_test, model_inception.predict(X_test).argmax(axis=1), zero_division=0))

print("Classification Report for Random Forest:") #Random Forest
print(classification_report(y_test, rf_pred, zero_division=0))
```

Code Snippet for Classification Report

Based on the performance comparison and classification reports, Inception-v3 turns out to be the best classification model among four models. It has achieved 65% accuracy which is highest of all and has a weighted F1-score of 0.61 which indicates better overall balance between precision and recall across all classes. It also shows reasonable performance on all three classes. Models like RestNet-50 and VGG-16 ignored classes 1 and 2 resulting in 0.00 F1-scores while Random Forest performed better than those two models but still struggled with class 2 as it only got 0.06 for F1-score. Random forest underperformed compared to Inception-v3, so it is the best model for this data in our use case.