Data Collection and Preprocessing Phase

Section	Description
Data Overview	**Dataset**: Medical images (retinal scans, OCT) with disease labels from sources like Kaggle and EyePACS. **Format & Labels**: JPEG/PNG/TIFF images with CSV metadata, containing disease or healthy labels. **Challenges**: Class imbalance and image quality issues, handled with preprocessing and augmentation.
Resizing	**Resizing**: Images are resized to a consistent dimension. **Uniformity**: Ensures consistent image size for model input. **Purpose**: Improves processing speed and model accuracy.
Normalization	**Normalization**: Pixel values are scaled to a range of [0, 1] for consistent model input. **Purpose**: Helps improve model convergence and accuracy during training.

Data Augmentation	**Data Augmentation**: Techniques like rotation, flipping, and zooming are applied to enhance dataset diversity. **Purpose**: Increases the variation in training data to prevent overfitting.

Date	02 October 2024
Team ID	739759
Project Title	OptiInsight - Revolutionizing Ophthalmic Care With Deep Learning For Predictive Eye Disease Analysis
Maximum Marks	6 Marks

Preprocessing Template the preprocessing steps involve several key stages. First, data collection is crucial, where you gather eye disease datasets such as retinal scans, OCT images, and other relevant data from public sources like Kaggle or EyePACS. Once the data is collected, it's important to clean it by addressing missing values, removing duplicates, and handling any corrupted image files.

	Benefit: Improves model generalization and robustness
Denoising	**Denoising**: Removes noise from medical images to improve clarity. **Purpose**: Enhances image quality for more accurate model predictions.
Edge Detection	**Edge Detection**: Identifies boundaries and key features in retinal images using techniques like Sobel or Canny. **Purpose**: Highlights critical structures, such as blood vessels and lesions, for disease detection. **Benefit**: Improves model focus on important image features for accurate diagnosis.
Color Space Conversion	**Color Space Conversion**: Converts images from RGB to grayscale for simpler analysis. **Purpose**: Reduces complexity while retaining key features. **Benefit**: Improves model efficiency and focus.
Image Cropping	**Image Cropping**: Crops irrelevant areas to focus on key features. **Purpose**: Highlights important regions like lesions or vessels. **Benefit**: Improves model accuracy with focused input data.
Batch Normalization	**Batch Normalization**: Normalizes activations within batches for stable training. **Purpose**: Speeds up convergence and reduces covariate shift. **Benefit**: Improves performance and reduces overfitting.

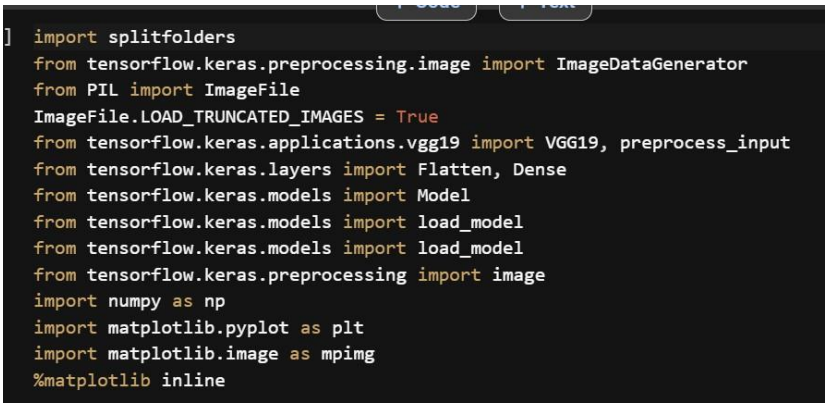
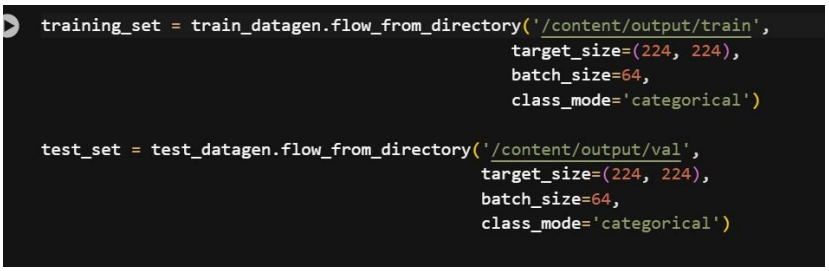
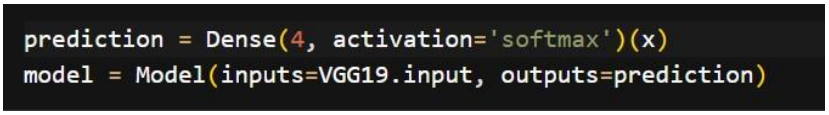
Data Preprocessing Code Screenshots

Data Collection	

```
import kagglehub

# Download latest version
path = kagglehub.dataset_download("gunavenkatdoddi/eye-diseases-classification")

print("Path to dataset files:", path)
```

Import the required library	
Apply Image Data Generator functionality to Trainset and Test set	
Pre-trained CNN model as a Feature Extractor	
Adding Dense Layer	
Train the model	

```
import splitfolders
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from PIL import ImageFile
ImageFile.LOAD_TRUNCATED_IMAGES = True
from tensorflow.keras.applications.vgg19 import VGG19, preprocess_input
from tensorflow.keras.layers import Flatten, Dense
from tensorflow.keras.models import Model
from tensorflow.keras.models import load_model
from tensorflow.keras.models import load_model
from tensorflow.keras.preprocessing import image
import numpy as np
import matplotlib.pyplot as plt
import matplotlib.image as mpimg
%matplotlib inline
```

```
training_set = train_datagen.flow_from_directory('/content/output/train',
                                                 target_size=(224, 224),
                                                 batch_size=64,
                                                 class_mode='categorical')

test_set = test_datagen.flow_from_directory('/content/output/val',
                                            target_size=(224, 224),
                                            batch_size=64,
                                            class_mode='categorical')
```

```
VGG19 = VGG19(input_shape=IMAGE_SIZE + [3], weights='imagenet', include_top=False)

# Downloading data from https://storage.googleapis.com/tensorflow/keras-applications/vgg
# 80134624/80134624 [==============================] - 1s 6us/step

for layer in VGG19.layers:
    layer.trainable = False

x = Flatten()(VGG19.output)
```

```
prediction = Dense(4, activation='softmax')(x)
model = Model(inputs=VGG19.input, outputs=prediction)
```

```
r = model.fit(
    training_set,
    validation_data=test_set,
    epochs=50,
    steps_per_epoch=len(training_set),
    validation_steps=len(test_set)
)
```

Save the Model