In [ ]:

In [ ]:

In [ ]:

In [1]:

```
from tensorflow import lite
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras import layers
import numpy as np
import pandas as pd
import random, os
import shutil
import matplotlib.pyplot as plt
from matplotlib.image import imread
# from keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.metrics import categorical_accuracy
from sklearn.model_selection import train_test_split
```

In [4]:

```
# Add an additional column, mapping to the type
df = pd.read_csv(r'train.csv')

diagnosis_dict_binary = {
    0: 'No_DR',
    1: 'DR',
    2: 'DR',
    3: 'DR',
    4: 'DR'
}

diagnosis_dict = {
    0: 'No_DR',
    1: 'Mild',
    2: 'Moderate',
    3: 'Severe',
    4: 'Proliferate_DR',
}


df['binary_type'] =  df['diagnosis'].map(diagnosis_dict_binary.get)
df['type'] = df['diagnosis'].map(diagnosis_dict.get)
df.head()
```

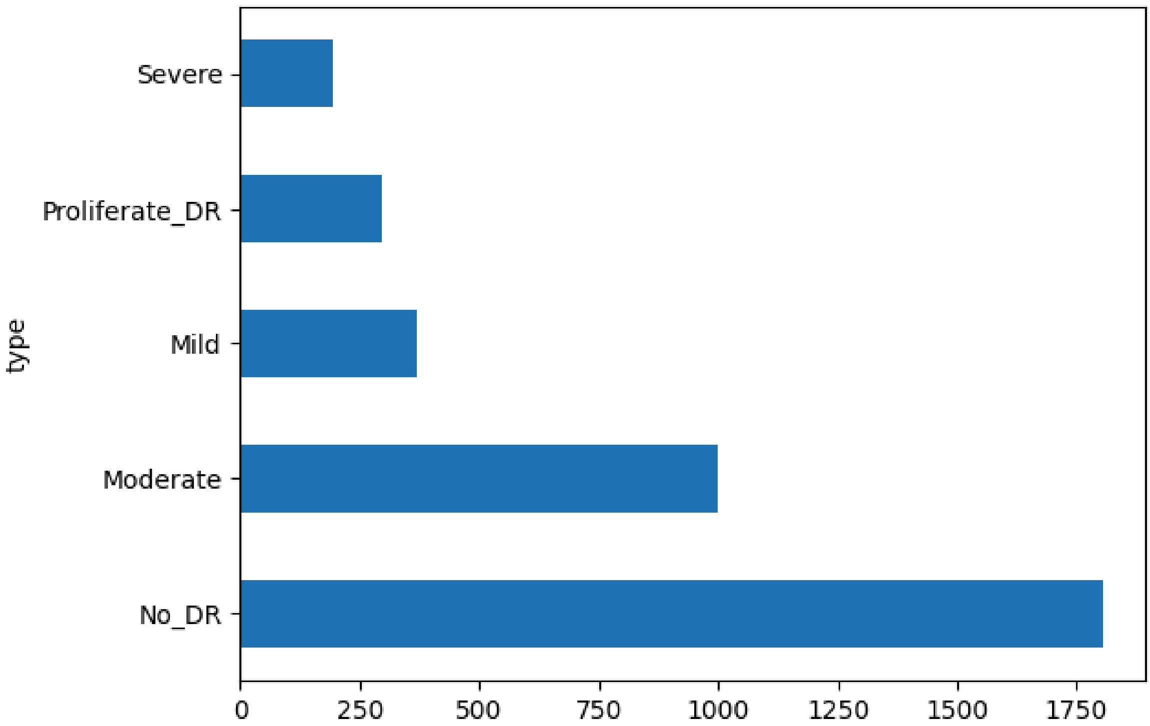
Out[4]:

	id_code	diagnosis	binary_type	type
0	000c1434d8d7	2	DR	Moderate
1	001639a390f0	4	DR	Proliferate_DR
2	0024cdab0c1e	1	DR	Mild
3	002c21358ce6	0	No_DR	No_DR
4	005b95c28852	0	No_DR	No_DR

In [5]:

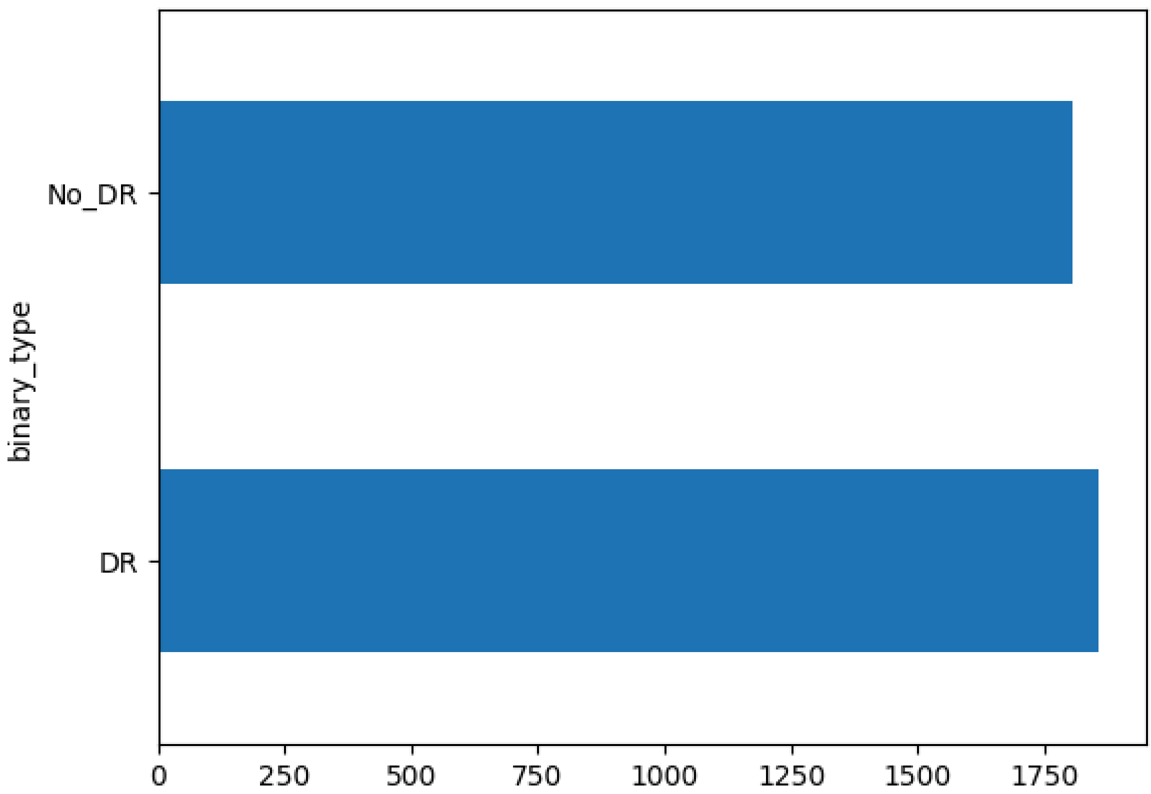
```
df['type'].value_counts().plot(kind='barh')
```

Out[5]: <Axes: ylabel='type'>



In [6]:
```
df['binary_type'].value_counts().plot(kind='barh')
```

Out[6]: <Axes: ylabel='binary_type'>



In [7]:
```
# Split into stratified train, val, and test sets
train_intermediate, val = train_test_split(df, test_size = 0.15, stratify = df['ty
train, test = train_test_split(train_intermediate, test_size = 0.15 / (1 - 0.15),

print(train['type'].value_counts(), '\n')
```

```
print(test['type'].value_counts(), '\n')
print(val['type'].value_counts(), '\n')
```

```
type
No_DR             1263
Moderate           699
Mild               258
Proliferate_DR     207
Severe             135
Name: count, dtype: int64

type
No_DR             271
Moderate          150
Mild               56
Proliferate_DR     44
Severe             29
Name: count, dtype: int64

type
No_DR             271
Moderate          150
Mild               56
Proliferate_DR     44
Severe             29
Name: count, dtype: int64
```

In [8]:

```
# Create working directories for train/val/test
base_dir = ''

train_dir = os.path.join(base_dir, 'train')
val_dir = os.path.join(base_dir, 'val')
test_dir = os.path.join(base_dir, 'test')

if os.path.exists(base_dir):
    shutil.rmtree(base_dir)

if os.path.exists(train_dir):
    shutil.rmtree(train_dir)
os.makedirs(train_dir)

if os.path.exists(val_dir):
    shutil.rmtree(val_dir)
os.makedirs(val_dir)

if os.path.exists(test_dir):
    shutil.rmtree(test_dir)
os.makedirs(test_dir)
```

In [12]:

```
# Copy images to respective working directory
src_dir = r'gaussian_filtered_images\gaussian_filtered_images'
for index, row in train.iterrows():
    diagnosis = row['type']
    binary_diagnosis = row['binary_type']
    id_code = row['id_code'] + ".png"
    srcfile = os.path.join(src_dir, diagnosis, id_code)
    dstfile = os.path.join(train_dir, binary_diagnosis)
    os.makedirs(dstfile, exist_ok = True)
    shutil.copy(srcfile, dstfile)
```

```
for index, row in val.iterrows():
    diagnosis = row['type']
    binary_diagnosis = row['binary_type']
    id_code = row['id_code'] + ".png"
    srcfile = os.path.join(src_dir, diagnosis, id_code)
    dstfile = os.path.join(val_dir, binary_diagnosis)
    os.makedirs(dstfile, exist_ok = True)
    shutil.copy(srcfile, dstfile)

for index, row in test.iterrows():
    diagnosis = row['type']
    binary_diagnosis = row['binary_type']
    id_code = row['id_code'] + ".png"
    srcfile = os.path.join(src_dir, diagnosis, id_code)
    dstfile = os.path.join(test_dir, binary_diagnosis)
    os.makedirs(dstfile, exist_ok = True)
    shutil.copy(srcfile, dstfile)
```

In [13]:

```
# Setting up ImageDataGenerator for train/val/test
from tensorflow.keras.preprocessing.image import ImageDataGenerator

train_path = 'train'
val_path = 'val'
test_path = 'test'

train_batches = ImageDataGenerator(rescale = 1./255).flow_from_directory(train_pat
val_batches = ImageDataGenerator(rescale = 1./255).flow_from_directory(val_path, t
test_batches = ImageDataGenerator(rescale = 1./255).flow_from_directory(test_path,
```

```
Found 2562 images belonging to 2 classes.
Found 550 images belonging to 2 classes.
Found 550 images belonging to 2 classes.
```

In [14]:

```
# Building the model

model = tf.keras.Sequential([
    # Layer 1: Convolutional Layer
    layers.Conv2D(8, (3,3), padding="valid", input_shape=(224,224,3), activation =
    layers.MaxPooling2D(pool_size=(2,2)),
    layers.BatchNormalization(),

      # Layer 2: Convolutional Layer
    layers.Conv2D(16, (3,3), padding="valid", activation = 'relu'),
    layers.MaxPooling2D(pool_size=(2,2)),
    layers.BatchNormalization(),
     # Layer 3: Convolutional Layer
    layers.Conv2D(32, (4,4), padding="valid", activation = 'relu'),
    layers.MaxPooling2D(pool_size=(2,2)),
    layers.BatchNormalization(),
     # Flatten the data
    layers.Flatten(),
    # Fully Connected Layer 1 (Dense Layer)
    layers.Dense(32, activation = 'relu'),
    layers.Dropout(0.15),
      # Output Layer
    layers.Dense(2, activation = 'softmax')
])

model.compile(optimizer=tf.keras.optimizers.Adam(learning_rate = 1e-5),
```

```
                 loss=tf.keras.losses.BinaryCrossentropy(),
                 metrics=['acc'])

history = model.fit(train_batches,
                    epochs=30,
                    validation_data=val_batches)
```

```
c:\Users\Gakps\anaconda3\envs\final\Lib\site-packages\keras\src\layers\convolutiona
l\base_conv.py:107: UserWarning: Do not pass an `input_shape`/`input_dim` argument 
to a layer. When using Sequential models, prefer using an `Input(shape)` object as 
the first layer in the model instead.
  super().__init__(activity_regularizer=activity_regularizer, **kwargs)
c:\Users\Gakps\anaconda3\envs\final\Lib\site-packages\keras\src\trainers\data_adapt
ers\py_dataset_adapter.py:121: UserWarning: Your `PyDataset` class should call `sup
er().__init__(**kwargs)` in its constructor. `**kwargs` can include `workers`, `use
_multiprocessing`, `max_queue_size`. Do not pass these arguments to `fit()`, as the
y will be ignored.
  self._warn_if_super_not_called()
```

```
Epoch 1/30
81/81 ━━━━━━━━━━━━━━━━━━━━ 22s 213ms/step - acc: 0.6760 - loss: 0.6624 - val_acc: 0.5073 - val_loss: 0.7078
Epoch 2/30
81/81 ━━━━━━━━━━━━━━━━━━━━ 16s 201ms/step - acc: 0.8887 - loss: 0.2837 - val_acc: 0.5073 - val_loss: 0.6908
Epoch 3/30
81/81 ━━━━━━━━━━━━━━━━━━━━ 16s 198ms/step - acc: 0.9176 - loss: 0.2487 - val_acc: 0.5800 - val_loss: 0.6391
Epoch 4/30
81/81 ━━━━━━━━━━━━━━━━━━━━ 17s 203ms/step - acc: 0.9211 - loss: 0.2313 - val_acc: 0.6709 - val_loss: 0.5522
Epoch 5/30
81/81 ━━━━━━━━━━━━━━━━━━━━ 17s 205ms/step - acc: 0.9264 - loss: 0.2149 - val_acc: 0.9109 - val_loss: 0.4127
Epoch 6/30
81/81 ━━━━━━━━━━━━━━━━━━━━ 17s 203ms/step - acc: 0.9243 - loss: 0.2265 - val_acc: 0.9273 - val_loss: 0.3060
Epoch 7/30
81/81 ━━━━━━━━━━━━━━━━━━━━ 17s 205ms/step - acc: 0.9377 - loss: 0.1973 - val_acc: 0.9327 - val_loss: 0.2471
Epoch 8/30
81/81 ━━━━━━━━━━━━━━━━━━━━ 17s 206ms/step - acc: 0.9393 - loss: 0.2032 - val_acc: 0.9291 - val_loss: 0.2146
Epoch 9/30
81/81 ━━━━━━━━━━━━━━━━━━━━ 17s 203ms/step - acc: 0.9392 - loss: 0.1934 - val_acc: 0.9436 - val_loss: 0.1846
Epoch 10/30
81/81 ━━━━━━━━━━━━━━━━━━━━ 17s 203ms/step - acc: 0.9430 - loss: 0.1751 - val_acc: 0.9436 - val_loss: 0.1824
Epoch 11/30
81/81 ━━━━━━━━━━━━━━━━━━━━ 17s 209ms/step - acc: 0.9509 - loss: 0.1598 - val_acc: 0.9455 - val_loss: 0.1726
Epoch 12/30
81/81 ━━━━━━━━━━━━━━━━━━━━ 16s 199ms/step - acc: 0.9443 - loss: 0.1675 - val_acc: 0.9509 - val_loss: 0.1704
Epoch 13/30
81/81 ━━━━━━━━━━━━━━━━━━━━ 16s 200ms/step - acc: 0.9551 - loss: 0.1487 - val_acc: 0.9418 - val_loss: 0.1754
Epoch 14/30
81/81 ━━━━━━━━━━━━━━━━━━━━ 17s 203ms/step - acc: 0.9526 - loss: 0.1592 - val_acc: 0.9473 - val_loss: 0.1709
Epoch 15/30
81/81 ━━━━━━━━━━━━━━━━━━━━ 16s 200ms/step - acc: 0.9546 - loss: 0.1557 - val_acc: 0.9491 - val_loss: 0.1625
Epoch 16/30
81/81 ━━━━━━━━━━━━━━━━━━━━ 17s 204ms/step - acc: 0.9554 - loss: 0.1470 - val_acc: 0.9473 - val_loss: 0.1619
Epoch 17/30
81/81 ━━━━━━━━━━━━━━━━━━━━ 17s 205ms/step - acc: 0.9544 - loss: 0.1392 - val_acc: 0.9491 - val_loss: 0.1666
Epoch 18/30
81/81 ━━━━━━━━━━━━━━━━━━━━ 16s 200ms/step - acc: 0.9626 - loss: 0.1297 - val_acc: 0.9509 - val_loss: 0.1599
Epoch 19/30
81/81 ━━━━━━━━━━━━━━━━━━━━ 16s 201ms/step - acc: 0.9612 - loss: 0.1341 - val_acc: 0.9509 - val_loss: 0.1577
Epoch 20/30
81/81 ━━━━━━━━━━━━━━━━━━━━ 17s 205ms/step - acc: 0.9638 - loss: 0.1314 - val_acc: 0.9473 - val_loss: 0.1569
```

```
Epoch 21/30
81/81 ━━━━━━━━━━━━━━━━━━━━ 16s 200ms/step - acc: 0.9532 - loss: 0.1344 - val_acc:
0.9382 - val_loss: 0.1667
Epoch 22/30
81/81 ━━━━━━━━━━━━━━━━━━━━ 17s 203ms/step - acc: 0.9663 - loss: 0.1155 - val_acc:
0.9473 - val_loss: 0.1550
Epoch 23/30
81/81 ━━━━━━━━━━━━━━━━━━━━ 17s 206ms/step - acc: 0.9607 - loss: 0.1188 - val_acc:
0.9455 - val_loss: 0.1556
Epoch 24/30
81/81 ━━━━━━━━━━━━━━━━━━━━ 17s 210ms/step - acc: 0.9607 - loss: 0.1179 - val_acc:
0.9527 - val_loss: 0.1505
Epoch 25/30
81/81 ━━━━━━━━━━━━━━━━━━━━ 17s 214ms/step - acc: 0.9618 - loss: 0.1184 - val_acc:
0.9473 - val_loss: 0.1581
Epoch 26/30
81/81 ━━━━━━━━━━━━━━━━━━━━ 17s 211ms/step - acc: 0.9661 - loss: 0.1081 - val_acc:
0.9509 - val_loss: 0.1512
Epoch 27/30
81/81 ━━━━━━━━━━━━━━━━━━━━ 17s 215ms/step - acc: 0.9622 - loss: 0.1162 - val_acc:
0.9473 - val_loss: 0.1507
Epoch 28/30
81/81 ━━━━━━━━━━━━━━━━━━━━ 17s 203ms/step - acc: 0.9726 - loss: 0.0996 - val_acc:
0.9527 - val_loss: 0.1516
Epoch 29/30
81/81 ━━━━━━━━━━━━━━━━━━━━ 17s 204ms/step - acc: 0.9652 - loss: 0.1010 - val_acc:
0.9509 - val_loss: 0.1482
Epoch 30/30
81/81 ━━━━━━━━━━━━━━━━━━━━ 16s 200ms/step - acc: 0.9761 - loss: 0.0851 - val_acc:
0.9491 - val_loss: 0.1484
```

In [45]:

```python
# Assuming `model` is your trained Keras model
model.save("CNN.h5")  # Save in HDF5 format
```

```
WARNING:absl:You are saving your model as an HDF5 file via `model.save()` or `kera
s.saving.save_model(model)`. This file format is considered legacy. We recommend us
ing instead the native Keras format, e.g. `model.save('my_model.keras')` or `keras.
saving.save_model(model, 'my_model.keras')`.
```

In [ ]:

```python
# Evaluate the model using the 'evaluate' method
loss, acc = model.evaluate(test_batches, verbose=1)

# Print results
print("Loss: ", loss)
print("Accuracy: ", acc)
```

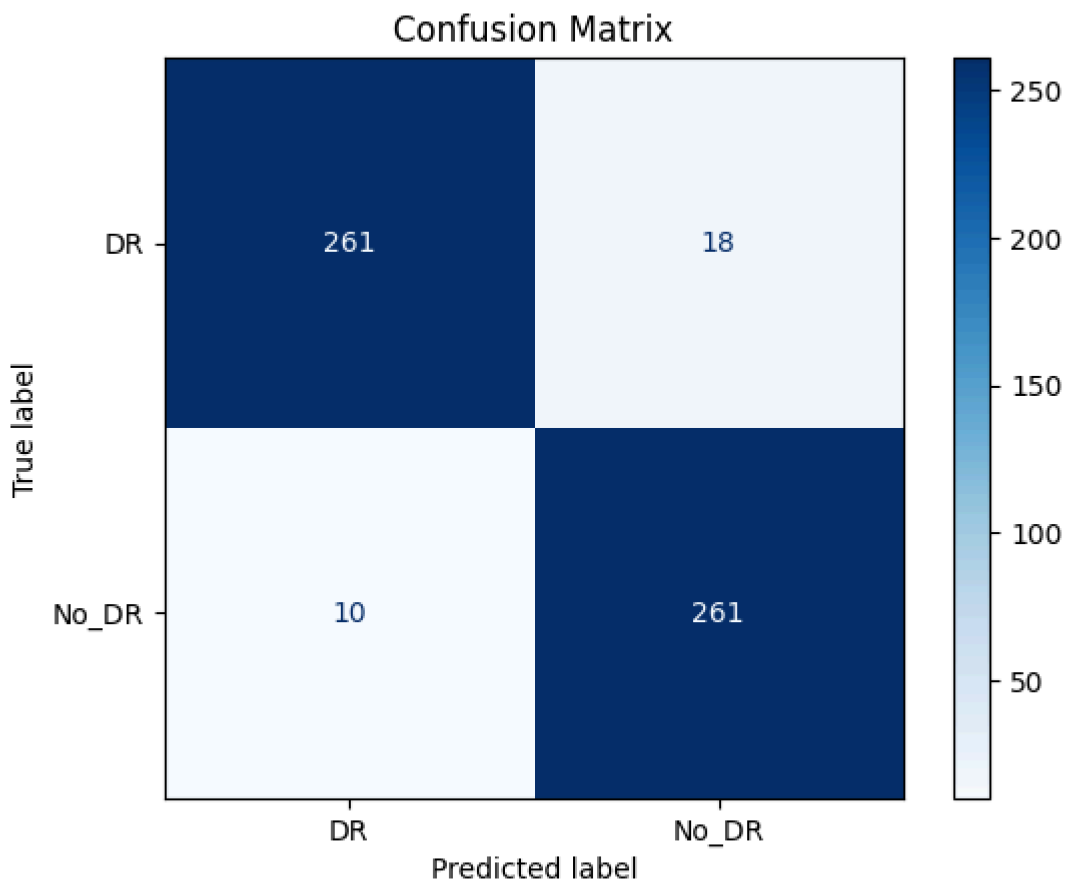
```
18/18 ━━━━━━━━━━━━━━━━━━━━ 3s 140ms/step - acc: 0.9433 - loss: 0.1720
Loss:  0.16484574973583221
Accuracy:  0.9490908980369568
```

In [16]:

```python
import numpy as np
import matplotlib.pyplot as plt
from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay

# Evaluate the model using the 'evaluate' method
loss, acc = model.evaluate(test_batches, verbose=1)

# Print results
print("Loss: ", loss)
```

```
print("Accuracy: ", acc)

# Generate predictions from the test data
y_pred = model.predict(test_batches)
y_pred_classes = np.argmax(y_pred, axis=1)  # Convert predictions to class labels

# Get true labels from test_batches
y_true = test_batches.classes  # Assuming test_batches is a generator with true la

# Generate the confusion matrix
cm = confusion_matrix(y_true, y_pred_classes)

# Plot the confusion matrix
disp = ConfusionMatrixDisplay(confusion_matrix=cm, display_labels=test_batches.cla
disp.plot(cmap=plt.cm.Blues)
plt.title("Confusion Matrix")
plt.show()
```

```
18/18 ━━━━━━━━━━━━━━━━━━━━ 3s 140ms/step - acc: 0.9433 - loss: 0.1720
Loss:  0.16484574973583221
Accuracy:  0.9490908980369568
18/18 ━━━━━━━━━━━━━━━━━━━━ 3s 141ms/step
```



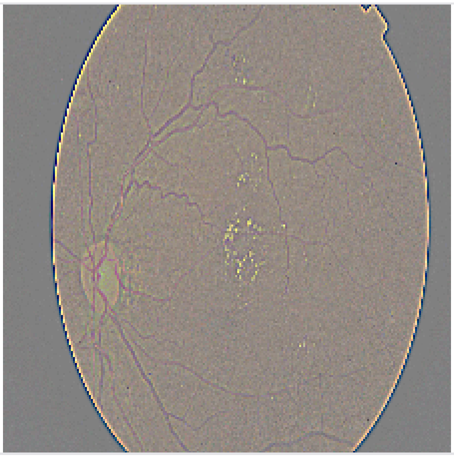
****Diabet Retinopathy Detection Section****

In [ ]:
```
import gradio as gr
import tensorflow as tf
from tensorflow.keras.preprocessing import image
import numpy as np
```

```python
# Load the trained model
model = tf.keras.models.load_model(r'CNN.h5')  # Replace with your model path

# Function to predict image
def predict_image(img):
    # Preprocess the image (resizing and scaling)
    img = img.resize((224, 224))
    img_array = np.array(img) / 255.0  # Rescale the image

    # Add batch dimension (as the model expects 4D input)
    img_array = np.expand_dims(img_array, axis=0)

    # Make prediction
    predictions = model.predict(img_array)

    # Get predicted class
    class_names = ['Diabetic Retinopathy', 'No Diabetic Retinopathy']  # Replace w
    predicted_class = class_names[np.argmax(predictions)]

    return predicted_class, predictions[0][np.argmax(predictions)]

# Build Gradio interface
iface = gr.Interface(fn=predict_image,
                     inputs=gr.Image(type="pil"),
                     outputs=[gr.Label(), gr.Textbox()],
                     live=True)

# Launch the Gradio app
iface.launch()
```

```
c:\Users\user\AppData\Local\Programs\Python\Python312\Lib\site-packages\tqdm\auto.p
y:21: TqdmWarning: IProgress not found. Please update jupyter and ipywidgets. See h
ttps://ipywidgets.readthedocs.io/en/stable/user_install.html
  from .autonotebook import tqdm as notebook_tqdm
WARNING:absl:Compiled the loaded model, but the compiled metrics have yet to be bui
lt. `model.compile_metrics` will be empty until you train or evaluate the model.
* Running on local URL:  http://127.0.0.1:7860
* To create a public link, set `share=True` in `launch()`.
```

img



Clear

output 0

Diabetic Retinopathy

output 1

Out[ ]:

```
1/1 ━━━━━━━━━━━━━━━━━━━━ 2s 2s/step
```

In [ ]:

In [ ]: