In [1]:

```
import os
import cv2
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import tensorflow.keras as keras
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.applications import ResNet50
from tensorflow.keras.layers import Dense, GlobalAveragePooling2D
from tensorflow.keras.models import Model
from sklearn.metrics import classification_report
```

In [2]:

```
dataset_dir = r"C:\Users\arnav\Desktop\Me\UST Global\Dataset\DRIVE"
image_size = (224, 224)
batch_size = 32
```

In [3]:

```
data_generator = ImageDataGenerator(
    rescale=1.0 / 255.0,
    rotation_range=20,
    width_shift_range=0.1,
    height_shift_range=0.1,
    shear_range=0.1,
    zoom_range=0.1,
    horizontal_flip=True,
    validation_split=0.2
)
```

In [4]:

```
train_generator = data_generator.flow_from_directory(
    dataset_dir,
    target_size=image_size,
    batch_size=batch_size,
    class_mode='categorical',
    subset='training'
)
```

```
Found 48 images belonging to 2 classes.
```

In [5]:

```
validation_generator = data_generator.flow_from_directory(
    dataset_dir,
    target_size=image_size,
    batch_size=batch_size,
    class_mode='categorical',
    subset='validation'
)
```

```
Found 12 images belonging to 2 classes.
```

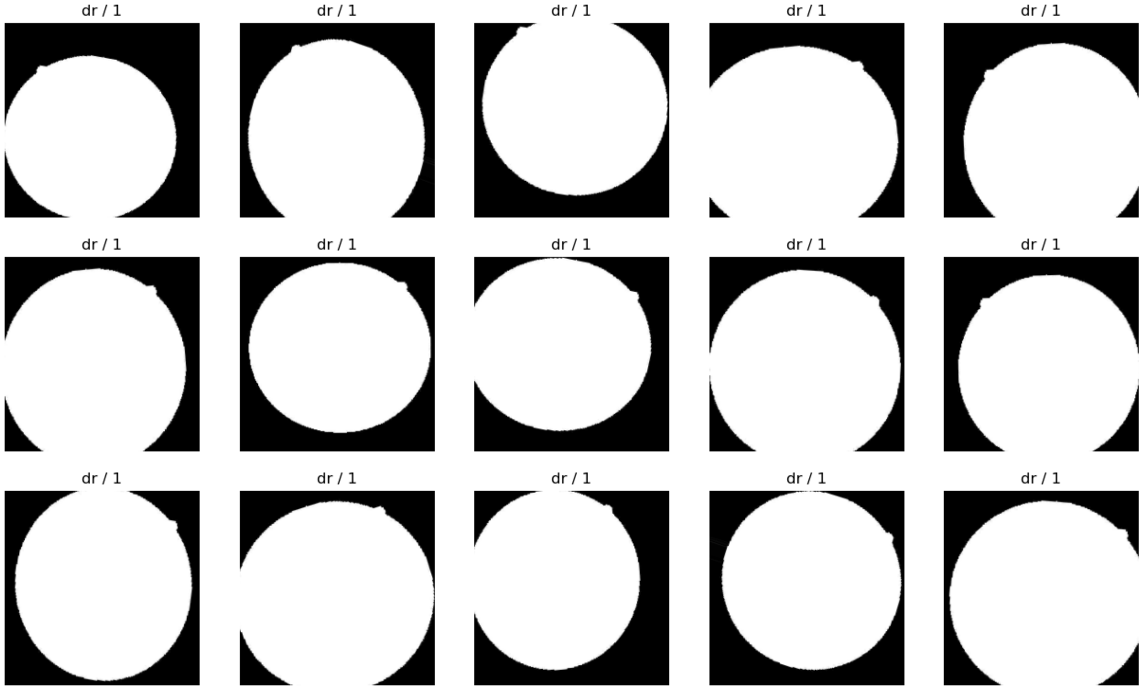
In [6]:

```
class_labels = {
    0: 'no dr',
    1: 'dr'
}
```

In [7]:

```
plt.figure(figsize=(16, 16))
j = 1
for i in np.random.randint(0, len(train_generator), 15):
    plt.subplot(5, 5, j)
    j += 1
    plt.imshow(train_generator[i][0][0], cmap="Greys")
    plt.axis('off')
    label = np.argmax(train_generator[i][1][0])
    plt.title('{} / {}'.format(class_labels[label], label))

plt.show()
```



In [8]:

```
base_model = ResNet50(weights='imagenet', include_top=False, input_shape=(224, 224, 3))
```

In [9]:

```
x = base_model.output
x = GlobalAveragePooling2D()(x)
x = Dense(256, activation='relu')(x)
predictions = Dense(2, activation='softmax')(x)
```

In [10]:

```
resnet_model = Model(inputs=base_model.input, outputs=predictions)
```

In [11]:

```
for layer in base_model.layers:
    layer.trainable = False
```

In [12]:

```
resnet_model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accurac
```

In [13]:

```
resnet_model.summary()
```

```
 conv3_block1_3_conv (Conv2D)   (None, 28, 28, 512)  66048       ['conv3
_block1_2_relu[0][0]']

 conv3_block1_0_bn (BatchNormal  (None, 28, 28, 512)  2048       ['conv3
_block1_0_conv[0][0]']
 ization)

 conv3_block1_3_bn (BatchNormal  (None, 28, 28, 512)  2048       ['conv3
_block1_3_conv[0][0]']
 ization)

 conv3_block1_add (Add)         (None, 28, 28, 512)  0           ['conv3
_block1_0_bn[0][0]',
                                                                  'conv3
_block1_3_bn[0][0]']

 conv3_block1_out (Activation)  (None, 28, 28, 512)  0           ['conv3
_block1_add[0][0]']

 conv3_block2_1_conv (Conv2D)   (None, 28, 28, 128)  65664       ['conv3
```

In [14]:

```
history = resnet_model.fit(
    train_generator,
    steps_per_epoch=train_generator.samples // batch_size,
    epochs=20,
    validation_data=validation_generator,
    validation_steps=validation_generator.samples // batch_size
)
```

```
Epoch 1/20
1/1 [==============================] - 7s 7s/step - loss: 0.8774 - accurac
y: 0.3750
Epoch 2/20
1/1 [==============================] - 1s 1s/step - loss: 0.4678 - accurac
y: 0.8125
Epoch 3/20
1/1 [==============================] - 3s 3s/step - loss: 1.4233 - accurac
y: 0.5625
Epoch 4/20
1/1 [==============================] - 3s 3s/step - loss: 0.8239 - accurac
y: 0.6562
Epoch 5/20
1/1 [==============================] - 1s 1s/step - loss: 0.5326 - accurac
y: 0.6250
Epoch 6/20
1/1 [==============================] - 3s 3s/step - loss: 0.4728 - accurac
y: 0.8125
Epoch 7/20
1/1 [==============================] - 3s 3s/step - loss: 0.7217 - accurac
y: 0.6875
Epoch 8/20
1/1 [==============================] - 1s 1s/step - loss: 0.6250 - accurac
y: 0.7500
Epoch 9/20
1/1 [==============================] - 1s 1s/step - loss: 0.6623 - accurac
y: 0.7500
Epoch 10/20
1/1 [==============================] - 1s 1s/step - loss: 0.5509 - accurac
y: 0.6875
Epoch 11/20
1/1 [==============================] - 1s 1s/step - loss: 0.4042 - accurac
y: 0.6250
Epoch 12/20
1/1 [==============================] - 3s 3s/step - loss: 0.6561 - accurac
y: 0.5938
Epoch 13/20
1/1 [==============================] - 1s 1s/step - loss: 0.8461 - accurac
y: 0.5625
Epoch 14/20
1/1 [==============================] - 3s 3s/step - loss: 0.6720 - accurac
y: 0.6562
Epoch 15/20
1/1 [==============================] - 3s 3s/step - loss: 0.5261 - accurac
y: 0.6875
Epoch 16/20
1/1 [==============================] - 1s 1s/step - loss: 0.5614 - accurac
y: 0.5000
Epoch 17/20
1/1 [==============================] - 3s 3s/step - loss: 0.4252 - accurac
y: 0.7188
Epoch 18/20
1/1 [==============================] - 3s 3s/step - loss: 0.4706 - accurac
y: 0.7500
Epoch 19/20
1/1 [==============================] - 3s 3s/step - loss: 0.4756 - accurac
y: 0.7812
Epoch 20/20
1/1 [==============================] - 1s 1s/step - loss: 0.3458 - accurac
y: 0.8750
```

In [15]:

```
test_dir = r"C:\Users\arnav\Desktop\Me\UST Global\Dataset\DRIVE\test"
test_generator = data_generator.flow_from_directory(
    test_dir,
    target_size=image_size,
    batch_size=batch_size,
    class_mode='categorical',
    shuffle=False
)
```

```
Found 20 images belonging to 2 classes.
```

In [16]:

```
test_loss, test_accuracy = resnet_model.evaluate(test_generator, verbose=1)
predictions = resnet_model.predict(test_generator)
predicted_labels = np.argmax(predictions, axis=1)
true_labels = test_generator.classes
print('Test loss:', test_loss)
print('Test accuracy:', test_accuracy)
```

```
1/1 [==============================] - 3s 3s/step - loss: 0.1494 - accurac
y: 1.0000
1/1 [==============================] - 3s 3s/step
Test loss: 0.14940036833286285
Test accuracy: 1.0
```

In [17]:

```
classification_report = classification_report(true_labels, predicted_labels)
print('Classification Report:\n', classification_report)
```

```
Classification Report:
               precision    recall  f1-score   support

           0       1.00      1.00      1.00        20

    accuracy                           1.00        20
   macro avg       1.00      1.00      1.00        20
weighted avg       1.00      1.00      1.00        20
```

In [22]:

```
def calculate_sensitivity_specificity(y_true, y_pred):
    tp = np.sum(np.logical_and(y_true == 1, y_pred == 1))
    tn = np.sum(np.logical_and(y_true == 0, y_pred == 0))
    fp = np.sum(np.logical_and(y_true == 0, y_pred == 1))
    fn = np.sum(np.logical_and(y_true == 1, y_pred == 0))
    sensitivity = tp / (tp + fn) if (tp + fn) != 0 else 0.0
    specificity = tn / (tn + fp) if (tn + fp) != 0 else 0.0

    return sensitivity, specificity

predicted_labels = np.argmax(predictions, axis=1)

sensitivity, specificity = calculate_sensitivity_specificity(true_labels, predicted_label

print('Specificity:', specificity)
```

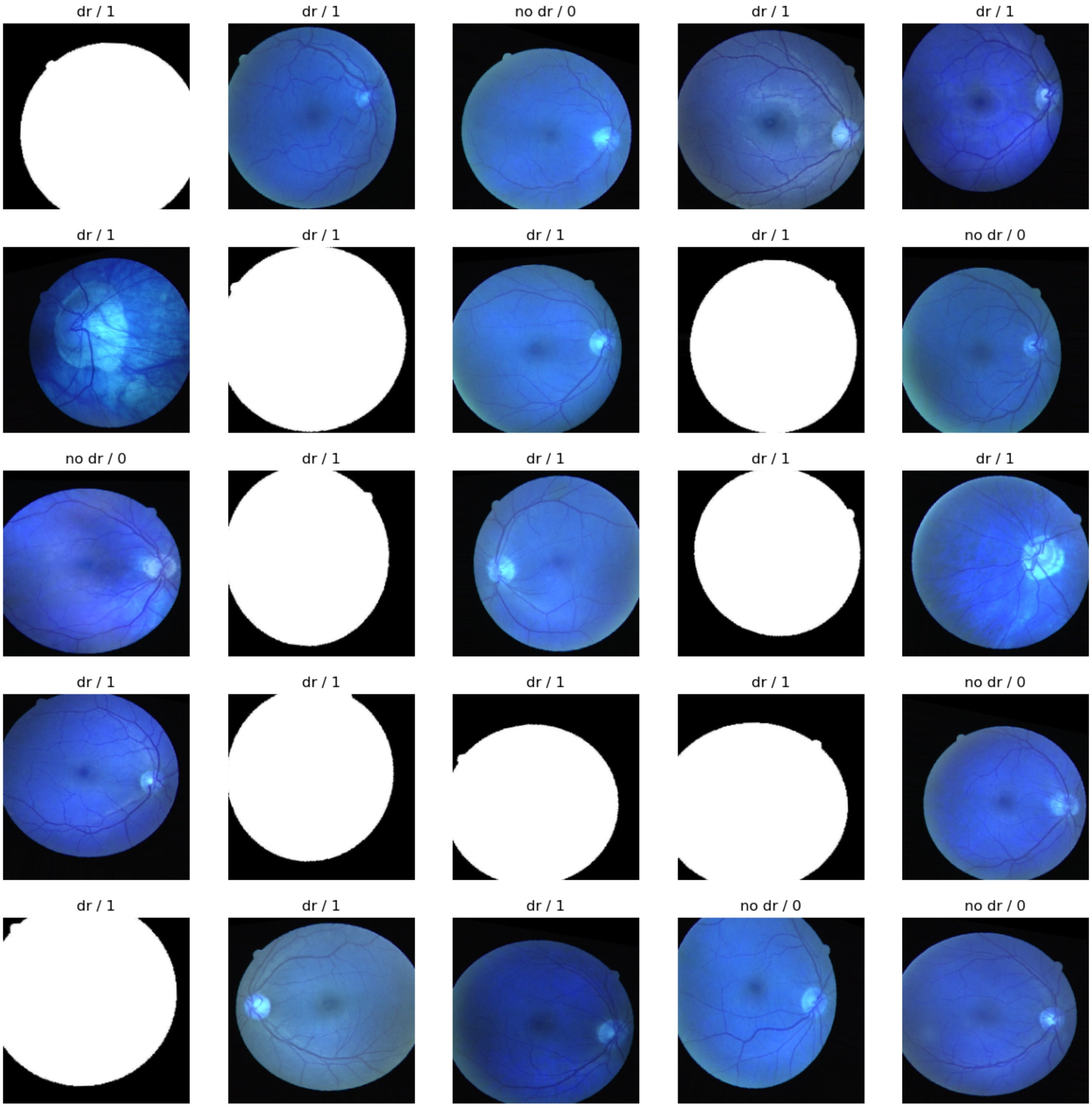
```
Specificity: 1.0
```

In [19]:

```
plt.figure(figsize=(16, 16))
j = 1
for batch in train_generator:
    images, labels = batch
    for i in range(len(images)):
        plt.subplot(5, 5, j)
        j += 1
        img = images[i]
        img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
        plt.imshow(img)
        plt.axis('off')
        label = np.argmax(labels[i])
        plt.title('{} / {}'.format(class_labels[label], label))

        if j > 25:
            break
    if j > 25:
        break

plt.show()
```



In [ ]: