In [25]:

```
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
```

In [26]:

```
dataset_dir = r"C:\Users\arnav\Desktop\Me\UST Global\Dataset\DRIVE"
```

In [27]:

```
image_size = (224,224)
batch_size = 32
```

In [28]:

```
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 [29]:

```
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 [30]:

```
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.
```

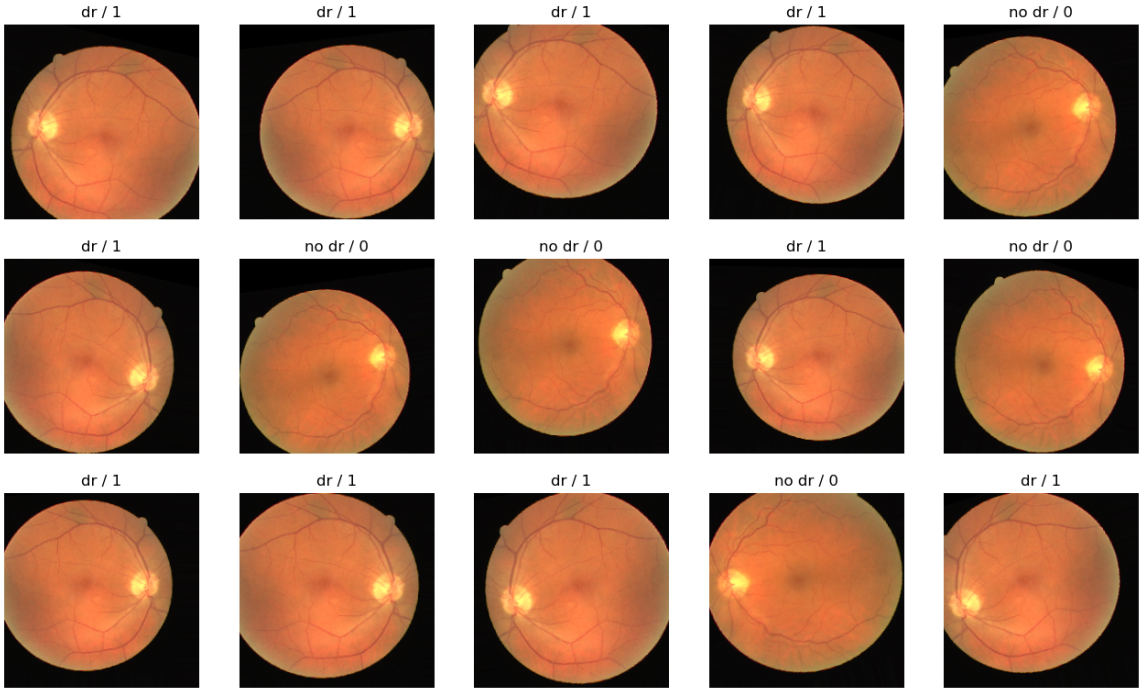
In [31]:

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

In [32]:

```
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 [33]:

```
cnn_model = keras.models.Sequential([
    keras.layers.Conv2D(filters=32, kernel_size=3, strides=(1, 1), padding='valid', activ
    keras.layers.MaxPooling2D(pool_size=(2, 2)),
    keras.layers.Conv2D(filters=64, kernel_size=3, strides=(1, 1), padding='valid', activ
    keras.layers.MaxPooling2D(pool_size=(2, 2)),
    keras.layers.Flatten(),
    keras.layers.Dense(256, activation='relu'),
    keras.layers.Dense(2, activation='softmax')
])
cnn_model.summary()
```

```
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d_2 (Conv2D)           (None, 222, 222, 32)      896

 max_pooling2d_2 (MaxPooling  (None, 111, 111, 32)     0
 2D)

 conv2d_3 (Conv2D)           (None, 109, 109, 64)      18496

 max_pooling2d_3 (MaxPooling  (None, 54, 54, 64)       0
 2D)

 flatten_1 (Flatten)         (None, 186624)            0

 dense_2 (Dense)             (None, 256)               47776000

 dense_3 (Dense)             (None, 2)                 514

=================================================================
Total params: 47,795,906
Trainable params: 47,795,906
Non-trainable params: 0
_________________________________________________________________
```

In [34]:

```
cnn_model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy']
```

In [35]:

```
history = cnn_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: 0.7172 - accurac
y: 0.3438
Epoch 2/20
1/1 [==============================] - 1s 1s/step - loss: 12.0498 - accura
cy: 0.6875
Epoch 3/20
1/1 [==============================] - 2s 2s/step - loss: 7.9359 - accurac
y: 0.6250
Epoch 4/20
1/1 [==============================] - 2s 2s/step - loss: 0.8469 - accurac
y: 0.5625
Epoch 5/20
1/1 [==============================] - 1s 1s/step - loss: 3.3644 - accurac
y: 0.4375
Epoch 6/20
1/1 [==============================] - 2s 2s/step - loss: 1.7658 - accurac
y: 0.3438
Epoch 7/20
1/1 [==============================] - 1s 1s/step - loss: 0.7735 - accurac
y: 0.6875
Epoch 8/20
1/1 [==============================] - 2s 2s/step - loss: 0.3504 - accurac
y: 0.7500
Epoch 9/20
1/1 [==============================] - 1s 1s/step - loss: 0.4151 - accurac
y: 0.6875
Epoch 10/20
1/1 [==============================] - 2s 2s/step - loss: 0.4558 - accurac
y: 0.6562
Epoch 11/20
1/1 [==============================] - 2s 2s/step - loss: 0.3958 - accurac
y: 0.7188
Epoch 12/20
1/1 [==============================] - 1s 1s/step - loss: 0.2442 - accurac
y: 0.8750
Epoch 13/20
1/1 [==============================] - 2s 2s/step - loss: 0.4141 - accurac
y: 0.7812
Epoch 14/20
1/1 [==============================] - 2s 2s/step - loss: 0.4196 - accurac
y: 0.7500
Epoch 15/20
1/1 [==============================] - 2s 2s/step - loss: 0.4379 - accurac
y: 0.7500
Epoch 16/20
1/1 [==============================] - 2s 2s/step - loss: 0.3923 - accurac
y: 0.7812
Epoch 17/20
1/1 [==============================] - 1s 1s/step - loss: 0.3828 - accurac
y: 0.8125
Epoch 18/20
1/1 [==============================] - 2s 2s/step - loss: 0.5417 - accurac
y: 0.6875
Epoch 19/20
1/1 [==============================] - 1s 1s/step - loss: 0.3877 - accurac
y: 0.8125
Epoch 20/20
1/1 [==============================] - 2s 2s/step - loss: 0.4257 - accurac
y: 0.6875
```

In [36]:

```
test_dir = r"C:\Users\arnav\Desktop\Me\UST Global\Dataset\DRIVE\test"
```

In [37]:

```
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 [38]:

```
test_images = []
test_labels = []
```

In [39]:

```
for i in range(len(test_generator)):
    batch_images, batch_labels = test_generator[i]
    test_images.append(batch_images)
    test_labels.append(batch_labels)
```

In [40]:

```
test_images = np.concatenate(test_images)
test_labels = np.concatenate(test_labels)
score = cnn_model.evaluate(test_images, test_labels, verbose=0)
print('Test loss:', score[0])
print('Test accuracy:', score[1])
```

```
Test loss: 0.517418384552002
Test accuracy: 0.949999988079071
```

In [44]:

```
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))

    # Calculate sensitivity and specificity
    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(cnn_model.predict(test_generator), axis=1)

sensitivity, specificity = calculate_sensitivity_specificity(test_generator.classes, pred

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

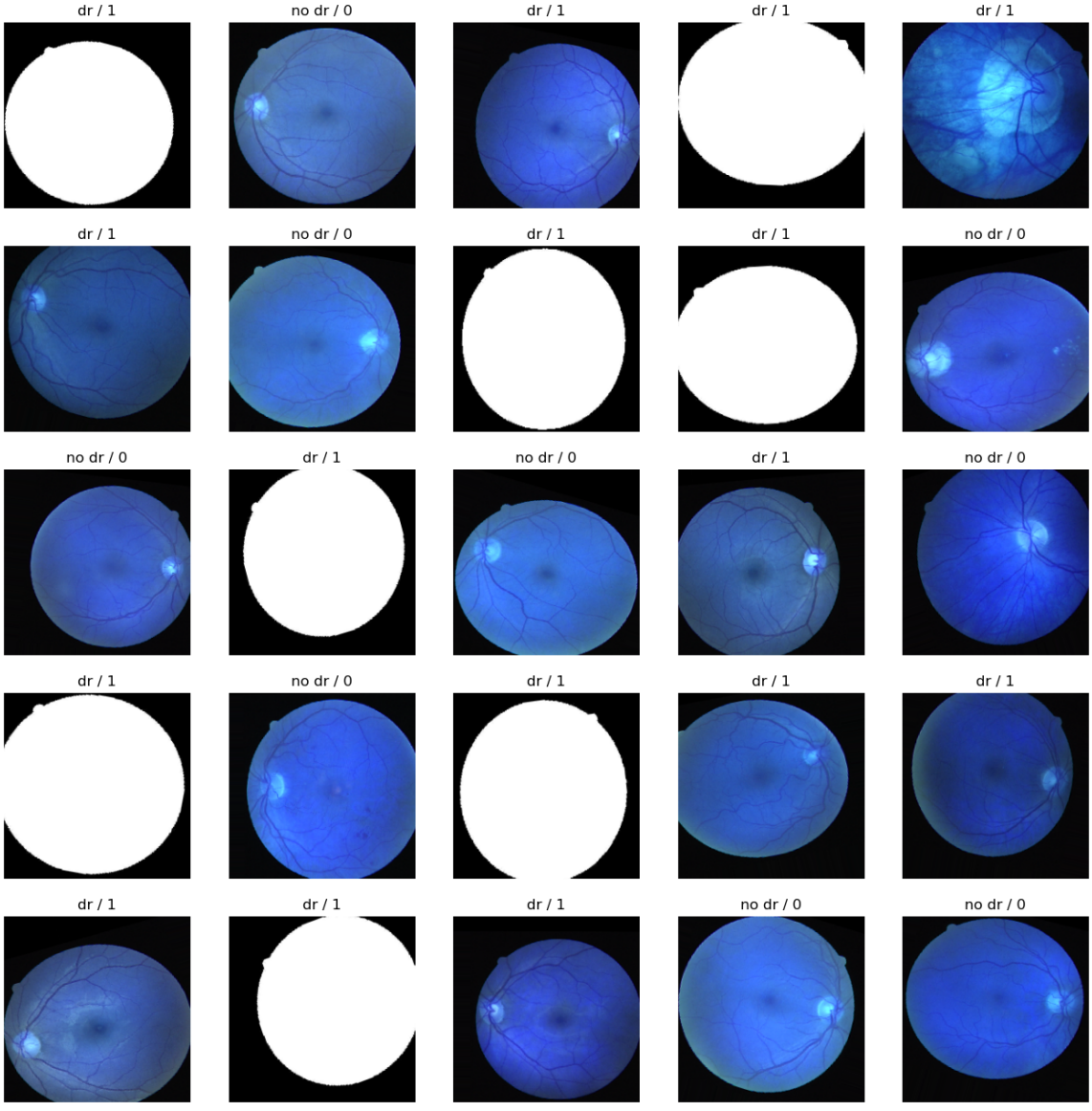
```
1/1 [==============================] - 1s 1s/step
Specificity: 0.95
```

In [42]:

```
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 [ ]:

```
```