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"
```

In [3]:

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

In [4]:

```
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 [5]:

```
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 [6]:

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

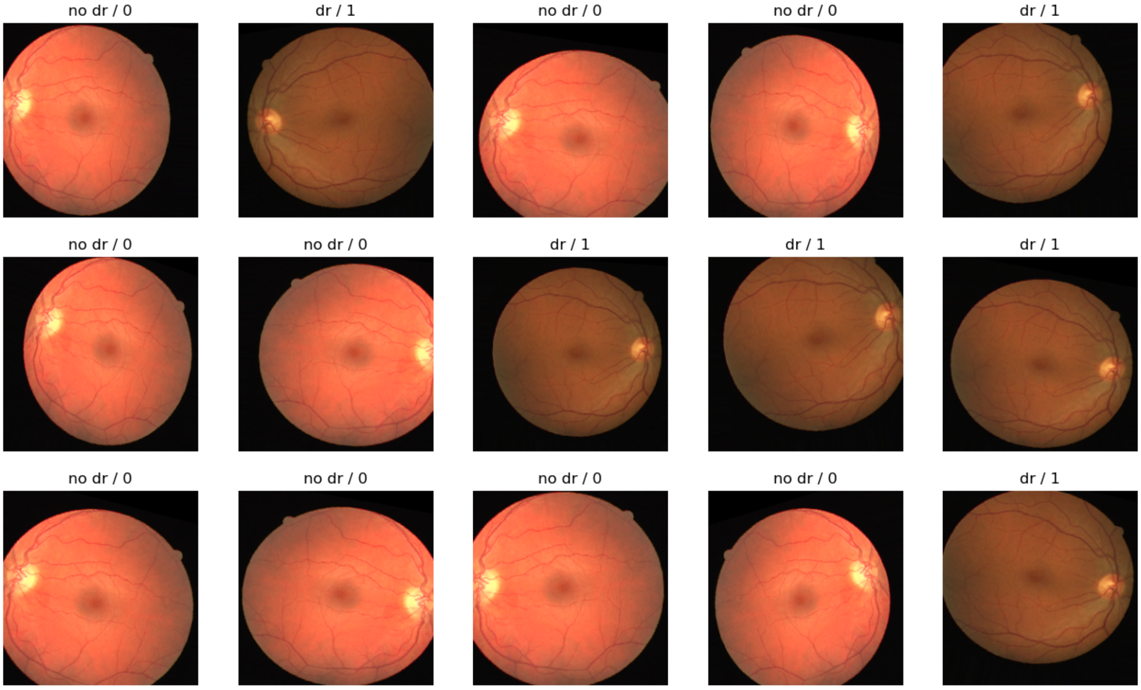
```
Found 12 images belonging to 2 classes.
```

In [7]:

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

In [8]:

```
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 [9]:

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

In [10]:

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

In [11]:

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

In [12]:

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

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

```
Model: "model"
________________________________________________________________________

__________________________
 Layer (type)                Output Shape         Param #     Connect
ed to
========================================================================
==========================
 input_1 (InputLayer)        [(None, 224, 224, 3  0           []
                             )]

 conv1_pad (ZeroPadding2D)   (None, 230, 230, 3)  0           ['input
_1[0][0]']

 conv1_conv (Conv2D)         (None, 112, 112, 64  9472        ['conv1
_pad[0][0]']

                             )

 conv1_bn (BatchNormalization)  (None, 112, 112, 64  256      ['conv1
_conv[0][0]']
```

In [13]:

```
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 [==============================] - 17s 17s/step - loss: 0.7005 - accur
acy: 0.4375
Epoch 2/20
1/1 [==============================] - 3s 3s/step - loss: 0.5649 - accurac
y: 0.8125
Epoch 3/20
1/1 [==============================] - 3s 3s/step - loss: 0.5907 - accurac
y: 0.7500
Epoch 4/20
1/1 [==============================] - 6s 6s/step - loss: 0.5077 - accurac
y: 0.6562
Epoch 5/20
1/1 [==============================] - 3s 3s/step - loss: 0.6159 - accurac
y: 0.7500
Epoch 6/20
1/1 [==============================] - 3s 3s/step - loss: 0.5625 - accurac
y: 0.7500
Epoch 7/20
1/1 [==============================] - 3s 3s/step - loss: 0.4057 - accurac
y: 0.7500
Epoch 8/20
1/1 [==============================] - 3s 3s/step - loss: 0.5565 - accurac
y: 0.5000
Epoch 9/20
1/1 [==============================] - 5s 5s/step - loss: 0.4730 - accurac
y: 0.6562
Epoch 10/20
1/1 [==============================] - 3s 3s/step - loss: 0.4148 - accurac
y: 0.6250
Epoch 11/20
1/1 [==============================] - 6s 6s/step - loss: 0.4019 - accurac
y: 0.7500
Epoch 12/20
1/1 [==============================] - 3s 3s/step - loss: 0.2864 - accurac
y: 0.8125
Epoch 13/20
1/1 [==============================] - 6s 6s/step - loss: 0.4248 - accurac
y: 0.7500
Epoch 14/20
1/1 [==============================] - 3s 3s/step - loss: 0.4000 - accurac
y: 0.7500
Epoch 15/20
1/1 [==============================] - 6s 6s/step - loss: 0.4101 - accurac
y: 0.6875
Epoch 16/20
1/1 [==============================] - 6s 6s/step - loss: 0.4343 - accurac
y: 0.6562
Epoch 17/20
1/1 [==============================] - 3s 3s/step - loss: 0.3166 - accurac
y: 0.7500
Epoch 18/20
1/1 [==============================] - 6s 6s/step - loss: 0.5103 - accurac
y: 0.5625
Epoch 19/20
1/1 [==============================] - 3s 3s/step - loss: 0.5277 - accurac
y: 0.5625
Epoch 20/20
1/1 [==============================] - 3s 3s/step - loss: 0.3659 - accurac
y: 0.7500
```

In [22]:

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

In [23]:

```
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 [24]:

```
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
```

```
1/1 [==============================] - 5s 5s/step - loss: 0.5466 - accurac
y: 0.9500
1/1 [==============================] - 5s 5s/step
```

In [25]:

```
print('Test loss:', test_loss)
print('Test accuracy:', test_accuracy)
```

```
Test loss: 0.5465558171272278
Test accuracy: 0.949999988079071
```

In [19]:

```
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
```

In [29]:

```
predicted_labels = np.argmax(predictions, axis=1)
sensitivity, specificity = calculate_sensitivity_specificity(true_labels, predicted_label
print('Specificity:', specificity)
```

```
Specificity: 0.961
```

In [21]:

```
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()
```