```
from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

```
cd /content/drive/MyDrive/Arthritis training Dataset
```

```
/content/drive/MyDrive/Knee-project
```

```
import tensorflow
print(tensorflow.__version__)
```

```
1.15.2
```

Data Preprocessing

```
import cv2,os
data_path='/content/drive/MyDrive/Knee-project/Arthritis training Dataset/'
categories=os.listdir(data_path)
labels=[i for i in range(len(categories))]

label_dict=dict(zip(categories,labels)) #empty dictionary
print(label_dict)
print(categories)
print(labels)
```

```
{'Normal': 0, 'Doubtful': 1, 'Mild': 2, 'Moderate': 3, 'Severe': 4}
['Normal', 'Doubtful', 'Mild', 'Moderate', 'Severe']
[0, 1, 2, 3, 4]
```

```
img_size=256
data=[]
label=[]

for category in categories:
    folder_path=os.path.join(data_path,category)
    img_names=os.listdir(folder_path)

    for img_name in img_names:
        img_path=os.path.join(folder_path,img_name)
        img=cv2.imread(img_path)
        try:
            gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
            resized=cv2.resize(gray,(img_size,img_size))
            #resizing the image  into 256 x 256, since we need a fixed common size for all the images in the dataset
            data.append(resized)
            label.append(label_dict[category])
            #appending the image and the label(categorized) into the list (dataset)
        except Exception as e:
            print('Exception:',e)
            #if any exception rasied, the exception will be printed here. And pass to the next image
```

Recale and assign catagorical labels

```
import numpy as np
data=np.array(data)/255.0
data=np.reshape(data,(data.shape[0],img_size,img_size,1))
label=np.array(label)
from keras.utils import np_utils
new_label=np_utils.to_categorical(label)
```

```
new_label.shape
```

```
(1650, 5)
```

CNN Model

```
data.shape
```

```
(1650, 256, 256, 1)
```

```
data.shape[1:]
```

```
(256, 256, 1)
```

```
from keras.models import Sequential
from keras.layers import Dense,Activation,Flatten,Dropout
from keras.layers import Conv2D,MaxPooling2D
from keras.callbacks import ModelCheckpoint

model=Sequential()

model.add(Conv2D(128,(3,3),input_shape=data.shape[1:]))
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size=(2,2)))
#The first CNN layer followed by Relu and MaxPooling layers

model.add(Conv2D(64,(3,3)))
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size=(2,2)))
#The second convolution layer followed by Relu and MaxPooling layers

model.add(Conv2D(32,(3,3)))
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size=(2,2)))
#The thrid convolution layer followed by Relu and MaxPooling layers

model.add(Flatten())
#Flatten layer to stack the output convolutions from 3rd convolution layer
model.add(Dropout(0.2))

model.add(Dense(128,activation='relu'))
#Dense layer of 128 neurons

model.add(Dropout(0.1))
model.add(Dense(64,activation='relu'))
#Dense layer of 64 neurons

model.add(Dense(5,activation='softmax'))
#The Final layer with two outputs for two categories

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

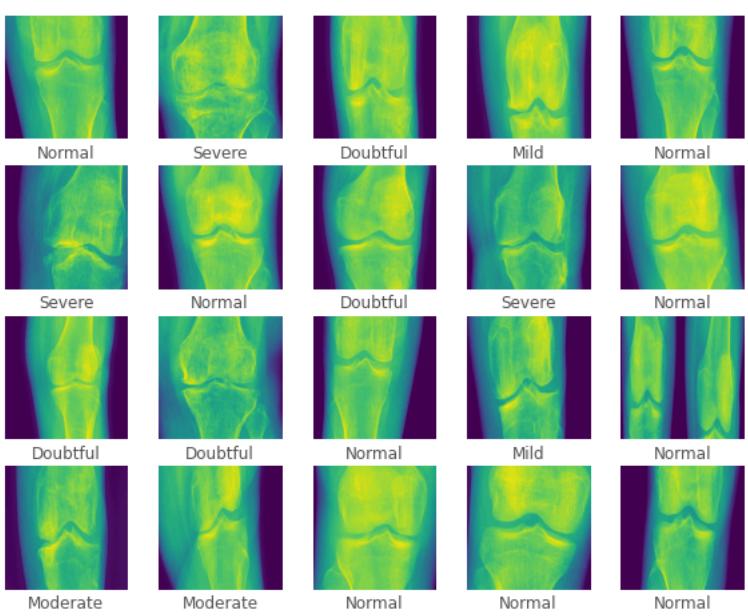
```
model.summary()
```

```
Model: "sequential_4"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_8 (Conv2D)            (None, 254, 254, 128)     1280
_________________________________________________________________
activation_8 (Activation)    (None, 254, 254, 128)     0
_________________________________________________________________
max_pooling2d_8 (MaxPooling2 (None, 127, 127, 128)     0
_________________________________________________________________
conv2d_9 (Conv2D)            (None, 125, 125, 64)      73792
_________________________________________________________________
activation_9 (Activation)    (None, 125, 125, 64)      0
_________________________________________________________________
max_pooling2d_9 (MaxPooling2 (None, 62, 62, 64)        0
_________________________________________________________________
conv2d_10 (Conv2D)           (None, 60, 60, 32)        18464
_________________________________________________________________
activation_10 (Activation)   (None, 60, 60, 32)        0
_________________________________________________________________
max_pooling2d_10 (MaxPooling (None, 30, 30, 32)        0
_________________________________________________________________
flatten_4 (Flatten)          (None, 28800)             0
_________________________________________________________________
dropout_4 (Dropout)          (None, 28800)             0
_________________________________________________________________
dense_8 (Dense)              (None, 128)               3686528
_________________________________________________________________
dropout_5 (Dropout)          (None, 128)               0
_________________________________________________________________
dense_9 (Dense)              (None, 64)                8256
_________________________________________________________________
dense_10 (Dense)             (None, 5)                 325
=================================================================
Total params: 3,788,645
Trainable params: 3,788,645
Non-trainable params: 0
```

Splitting data into traning and testing

```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(data,new_label,test_size=0.1)
```

```
import matplotlib.pyplot as plt
plt.figure(figsize=(10,10))
for i in range(20):
    plt.subplot(5,5,i+1)
    plt.xticks([])
    plt.yticks([])
    plt.grid(False)
    plt.imshow(np.squeeze(x_test[i]))
    plt.xlabel(categories[np.argmax(y_test[i])])
plt.show()
```



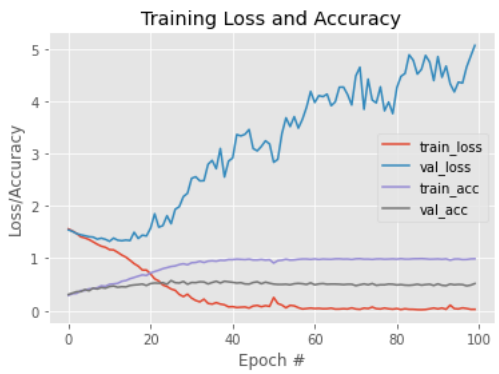

```
history=model.fit(x_train,y_train,epochs=100,validation_split=0.2)
```

```
Epoch 90/100
1188/1188 [==============================] - 6s 5ms/step - loss: 0.0508 - accuracy: 0.9857 - val_loss: 4.3948 - val_accuracy: 0.4
Epoch 91/100
1188/1188 [==============================] - 6s 5ms/step - loss: 0.0346 - accuracy: 0.9891 - val_loss: 4.8526 - val_accuracy: 0.5
Epoch 92/100
1188/1188 [==============================] - 6s 5ms/step - loss: 0.0512 - accuracy: 0.9832 - val_loss: 4.4578 - val_accuracy: 0.4
Epoch 93/100
1188/1188 [==============================] - 6s 5ms/step - loss: 0.0269 - accuracy: 0.9899 - val_loss: 4.6724 - val_accuracy: 0.4
Epoch 94/100
1188/1188 [==============================] - 6s 5ms/step - loss: 0.1069 - accuracy: 0.9655 - val_loss: 4.3326 - val_accuracy: 0.5
Epoch 95/100
1188/1188 [==============================] - 6s 5ms/step - loss: 0.0401 - accuracy: 0.9865 - val_loss: 4.1785 - val_accuracy: 0.4
Epoch 96/100
1188/1188 [==============================] - 6s 5ms/step - loss: 0.0340 - accuracy: 0.9882 - val_loss: 4.3658 - val_accuracy: 0.5
Epoch 97/100
1188/1188 [==============================] - 6s 5ms/step - loss: 0.0539 - accuracy: 0.9773 - val_loss: 4.3475 - val_accuracy: 0.4
Epoch 98/100
1188/1188 [==============================] - 6s 5ms/step - loss: 0.0422 - accuracy: 0.9790 - val_loss: 4.6538 - val_accuracy: 0.4
Epoch 99/100
1188/1188 [==============================] - 6s 5ms/step - loss: 0.0263 - accuracy: 0.9891 - val_loss: 4.8608 - val_accuracy: 0.4
Epoch 100/100
```

```
model.save('model.h5')
```

```
from matplotlib import pyplot as plt
```

```
# plot the training loss and accuracy
N = 100 #number of epochs
plt.style.use("ggplot")
plt.figure()
plt.plot(np.arange(0, N), history.history["loss"], label="train_loss")
plt.plot(np.arange(0, N), history.history["val_loss"], label="val_loss")
plt.plot(np.arange(0, N), history.history["accuracy"], label="train_acc")
plt.plot(np.arange(0, N), history.history["val_accuracy"], label="val_acc")
plt.title("Training Loss and Accuracy")
plt.xlabel("Epoch #")
plt.ylabel("Loss/Accuracy")
plt.legend(loc="center right")
plt.savefig("CNN_Model")
```




```
val_loss, val_accuracy= model.evaluate(x_test, y_test, verbose=0)
print("test loss:", val_loss,'%')
print("test accuracy:", val_accuracy,"%")
```

```
test loss: 5.4802391427935975 %
test accuracy: 0.4909090995788574 %
```

```
X = 32

img_size = 256

img_single = x_test[X]
img_single = cv2.resize(img_single, (img_size, img_size))
img_single = (np.expand_dims(img_single, 0))
img_single = img_single.reshape(img_single.shape[0],256,256,1)

predictions_single = model.predict(img_single)
print('A.I predicts:',categories[np.argmax(predictions_single)])
print("Correct prediction for label",np.argmax(y_test[X]),'is',categories[np.argmax(y_test[X])])
plt.imshow(np.squeeze(img_single))
plt.grid(False)
plt.show()
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-1-1c8281d4ddd7> in <cell line: 5>()
      3 img_size = 256
      4
----> 5 img_single = x_test[X]
      6 img_single = cv2.resize(img_single, (img_size, img_size))
      7 img_single = (np.expand_dims(img_single, 0))

NameError: name 'x_test' is not defined
```

Start coding or generate with AI.