Machine Learning Internship in collaboration with Start Tech Academy

TASK 2: Build a Deep learning model to detect if osteoarthritis is present or not in a given knee X-ray image.

In [1]:
```
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Activation,Flatten, Dense, Dropout, Conv2D, MaxPooling2D
import tensorflow as tf
import numpy as np
import matplotlib.pyplot as plt
import os
import cv2
import random
import pickle
from tqdm import tqdm
```

NOTE: all images in the dataset is 224 X 224

In [3]:
```
datadir="D:/Deep Learning/train"
categories=["Normal","Osteoarthritis"]

IMG_SIZE = 60
```

In [5]:
```
training_data=[]

def create_training_data():
    for category in categories:

        path = os.path.join(datadir,category)
        class_num = categories.index(category)

        for img in tqdm(os.listdir(path)):
            try:
                img_array = cv2.imread(os.path.join(path,img) ,cv2.IMREAD_GRAYSCALE)
                new_array = cv2.resize(img_array, (IMG_SIZE, IMG_SIZE))
                training_data.append([new_array, class_num])
            except Exception as e:
                pass


create_training_data()


print(len(training_data))
```

```
100%|██████████| 810/810 [00:01<00:00, 690.15it/s]
100%|██████████| 1540/1540 [00:02<00:00, 735.05it/s]
2350
```

In [6]:
```
random.shuffle(training_data)
```

In [7]:
```
X=[]
y=[]
for features,labels in training_data:
    X.append(features)
    y.append(labels)
X=np.array(X).reshape(-1,IMG_SIZE,IMG_SIZE,1)
y=np.array(y)
X.shape   #(2350, 60, 60, 1)

y.shape  #(2350,)
```

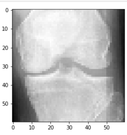
Out[7]:
```
(2350,)
```

In [8]:
```
pickle_out=open("X.pickle","wb")
pickle.dump(X,pickle_out)
pickle_out.close()

pickle_out=open("y.pickle","wb")
pickle.dump(y,pickle_out)
pickle_out.close()
```

In [9]:
```
pickle_in=open("X.pickle","rb")
X=pickle.load(pickle_in)
```

In [10]:
```
plt.imshow(X[0],cmap="gray")
plt.show()
```



In [16]:
```
X[0]
```

Out[16]:
```
array([[[0.58431373],
        [0.64313725],
        [0.69019608],
        ...,
        [0.25882353],
        [0.23529412],
        [0.21568627]],

       [[0.56470588],
        [0.6       ],
        [0.66666667],
        ...,
        [0.25490196],
        [0.23921569],
        [0.21568627]],

       [[0.54509804],
        [0.58823529],
        [0.62745098],
        ...,
        [0.25098039],
        [0.22352941],
        [0.20392157]],

       ...,

       [[0.12941176],
        [0.16862745],
        [0.19215686],
        ...,
        [0.54901961],
        [0.5372549 ],
        [0.44705882]],

       [[0.11764706],
        [0.17254902],
        [0.18823529],
        ...,
        [0.5254902 ],
        [0.57647059],
        [0.37254902]],

       [[0.10980392],
        [0.17254902],
        [0.19215686],
        ...,
        [0.61568627],
        [0.46666667],
        [0.36470588]]])
```

In [11]:
```
X=pickle.load(open("X.pickle","rb"))
y=pickle.load(open("y.pickle","rb"))
```

Neural Network Architecture:

In [12]:
```
X=X/255.0    #normalization

model=Sequential()
model.add(Conv2D(32,(3,3), input_shape=X.shape[1:]))
model.add(Activation("relu"))
model.add(MaxPooling2D(pool_size=(2,2)))


model.add(Conv2D(32,(3,3)))
model.add(Activation("relu"))
model.add(MaxPooling2D(pool_size=(2,2)))


model.add(Flatten())
model.add(Dense(32))
model.add(Dense(32))

model.add(Dense(32))



model.add(Dense(1))
model.add(Activation("sigmoid"))


model.compile(loss="binary_crossentropy",optimizer="adam", metrics=['accuracy'])


model.fit(X,y,batch_size=32,epochs=10)
```

```
Epoch 1/10
74/74 [==============================] - 8s 88ms/step - loss: 0.5960 - accuracy: 0.6494
Epoch 2/10
74/74 [==============================] - 7s 89ms/step - loss: 0.5440 - accuracy: 0.6702
Epoch 3/10
74/74 [==============================] - 6s 86ms/step - loss: 0.4910 - accuracy: 0.7298
Epoch 4/10
74/74 [==============================] - 7s 89ms/step - loss: 0.4447 - accuracy: 0.7745
Epoch 5/10
74/74 [==============================] - 7s 94ms/step - loss: 0.3912 - accuracy: 0.8094
Epoch 6/10
74/74 [==============================] - 6s 87ms/step - loss: 0.3522 - accuracy: 0.8336
Epoch 7/10
74/74 [==============================] - 7s 88ms/step - loss: 0.3001 - accuracy: 0.8596
Epoch 8/10
74/74 [==============================] - 6s 87ms/step - loss: 0.3152 - accuracy: 0.8549
Epoch 9/10
74/74 [==============================] - 7s 89ms/step - loss: 0.2591 - accuracy: 0.8817
Epoch 10/10
74/74 [==============================] - 7s 88ms/step - loss: 0.2618 - accuracy: 0.8804
```

Out[12]:
```
<keras.callbacks.History at 0x1c707c6d0a0>
```

Evaluation on Testing Dataset

In [13]:
```
datadir="D:/Deep Learning/test"
categories=["Normal","Osteoarthritis"]
for i in categories:
    path=os.path.join(datadir,i)
    for img in os.listdir(path):
        img_array=cv2.imread(os.path.join(path,img),cv2.IMREAD_GRAYSCALE)


IMG_SIZE=60
new_array=cv2.resize(img_array,(IMG_SIZE,IMG_SIZE))

testing_data=[]

def create_testing_data():
    for category in categories:

        path = os.path.join(datadir,category)
        class_num = categories.index(category)

        for img in tqdm(os.listdir(path)):
            try:
                img_array = cv2.imread(os.path.join(path,img) ,cv2.IMREAD_GRAYSCALE)
                new_array = cv2.resize(img_array, (IMG_SIZE, IMG_SIZE))
                testing_data.append([new_array, class_num])
            except Exception as e:
                pass


create_testing_data()


print(len(testing_data))
```

```
100%|██████████| 569/569 [00:00<00:00, 667.37it/s]
100%|██████████| 276/276 [00:00<00:00, 578.13it/s]
845
```

In [15]:
```
random.shuffle(testing_data)

X_test=[]
y_test=[]
for features,labels in testing_data:
    X_test.append(features)
    y_test.append(labels)
X_test=np.array(X_test).reshape(-1,IMG_SIZE,IMG_SIZE,1)
y_test=np.array(y_test)

X_test=X_test/255.0

model.fit(X_test,y_test,epochs=10)
```

```
Epoch 1/10
27/27 [==============================] - 2s 86ms/step - loss: 0.5159 - accuracy: 0.7314
Epoch 2/10
27/27 [==============================] - 2s 84ms/step - loss: 0.4806 - accuracy: 0.7444
Epoch 3/10
27/27 [==============================] - 2s 84ms/step - loss: 0.4564 - accuracy: 0.7822
Epoch 4/10
27/27 [==============================] - 2s 84ms/step - loss: 0.4554 - accuracy: 0.7870
Epoch 5/10
27/27 [==============================] - 2s 90ms/step - loss: 0.4767 - accuracy: 0.7586
Epoch 6/10
27/27 [==============================] - 2s 85ms/step - loss: 0.4132 - accuracy: 0.8107
Epoch 7/10
27/27 [==============================] - 2s 85ms/step - loss: 0.4787 - accuracy: 0.7609
Epoch 8/10
27/27 [==============================] - 2s 87ms/step - loss: 0.4073 - accuracy: 0.8189
Epoch 9/10
27/27 [==============================] - 2s 87ms/step - loss: 0.3974 - accuracy: 0.8355
Epoch 10/10
27/27 [==============================] - 2s 88ms/step - loss: 0.3866 - accuracy: 0.8308
```

Out[15]:
```
<keras.callbacks.History at 0x1c70810c090>
```

Resuts:

Accuracy of the model with training data: **88.04%**

Accuracy of the model with testing data: **83.08%**

In [ ]:
```
```