Begin by creating the ipynb file in the same parent folder where the downloaded data set is kept. The CNN model should have the following layers: ● Input layer ● Convolutional layer 1 with 32 filters of kernel size[5,5] ● Pooling layer 1 with pool size[2,2] and stride 2 ● Convolutional layer 2 with 64 filters of kernel size[5,5] ● Pooling layer 2 with pool size[2,2] and stride 2 ● Dense layer whose output size is fixed in the hyper parameter: fc_size=32 ● Dropout layer with dropout probability 0.4 Predict the class by doing a softmax on the output of the dropout layers. This should be followed by training and evaluation: ● For the training step, define the loss function and minimize it ● For the evaluation step, calculate the accuracy Run the program for 100, 200, and 300 iterations, respectively. Follow this by a report on the final accuracy and loss on the evaluation data.

In [1]:
```
#import all the require labraries
```

In [2]:
```
import tensorflow as tf
from tensorflow import keras
from keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras import models, layers
```

```
C:\Users\aditya\Anaconda3\lib\site-packages\scipy\__init__.py:146: UserWarning: A NumPy version >=1.16.5 and <
1.23.0 is required for this version of SciPy (detected version 1.23.2
  warnings.warn(f"A NumPy version >={np_minversion} and <{np_maxversion}"
```

In [3]:
```
dataflow = ImageDataGenerator(rescale = 1.0 / 255.0 )
```

In [4]:
```
#import training data from file
```

In [5]:
```
train = dataflow.flow_from_directory('E:/downloads/TENSER FLOW AND KERAS DEEP LEARNING/data/train', class_mode =
```

```
Found 40 images belonging to 2 classes.
```

In [6]:
```
#import test data from file
```

In [7]:
```
test = dataflow.flow_from_directory('E:/downloads/TENSER FLOW AND KERAS DEEP LEARNING/data/test', class_mode = '
```

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

In [8]:
```
#build model according to
#Input layer • Convolutional layer 1 with 32 filters of kernel size[5,5]
#Pooling layer 1 with pool size[2,2] and stride
#Convolutional layer 2 with 64 filters of kernel size[5,5]
#Pooling layer 2 with pool size[2,2] and stride 2
#Dense layer whose output size is fixed in the hyper parameter: fc_size=32
#Dropout layer with dropout probability 0.4 Predict the class by doing a softmax on the output of the dropout la
#
```

In [9]:
```
model = models.Sequential()
model.add( layers.Conv2D( 32, (5, 5 ), activation = 'relu', padding = 'same', input_shape = (256,  256 , 3 )))
model.add(layers.MaxPooling2D(2,2))
model.add(layers.Conv2D(64, (5, 5 ), activation = 'relu'))
model.add(layers.MaxPooling2D((2,2)))
model.add(layers.Dropout(0.4))
model.add(layers.Flatten())
model.add(layers.Dense(32, activation = 'relu'))
model.add(layers.Dense(2, activation = 'softmax'))
```

In [10]:
```
#For the training step, define the loss function and minimize it
```

In [11]:
```
sgd_opt = tf.keras.optimizers.SGD(lr = 0.001)
```

```
C:\Users\aditya\Anaconda3\lib\site-packages\keras\optimizers\optimizer_v2\gradient_descent.py:111: UserWarnin
g: The `lr` argument is deprecated, use `learning_rate` instead.
  super().__init__(name, **kwargs)
```

In [12]:
```
#compiling model
```

In [13]:
```
history=model.compile( optimizer = sgd_opt, loss = 'binary_crossentropy', metrics =  ['accuracy'])
```

In [14]:
```
#model train for 100 epochs
```

In [15]:
```
history =model.fit(train, validation_data = test, epochs =10)
```

```
Epoch 1/10
2/2 [==============================] - 38s 19s/step - loss: 0.6869 - accuracy: 0.5750 - val_loss: 0.6863 - val
_accuracy: 0.7000
Epoch 2/10
2/2 [==============================] - 16s 5s/step - loss: 0.6764 - accuracy: 0.5750 - val_loss: 0.6936 - val_
accuracy: 0.5000
Epoch 3/10
2/2 [==============================] - 16s 13s/step - loss: 0.6955 - accuracy: 0.5000 - val_loss: 0.6909 - val
_accuracy: 0.5500
Epoch 4/10
2/2 [==============================] - 17s 5s/step - loss: 0.6823 - accuracy: 0.6000 - val_loss: 0.6898 - val_
accuracy: 0.6000
Epoch 5/10
2/2 [==============================] - 20s 15s/step - loss: 0.7148 - accuracy: 0.5250 - val_loss: 0.6944 - val
_accuracy: 0.5000
Epoch 6/10
2/2 [==============================] - 18s 5s/step - loss: 0.6733 - accuracy: 0.5750 - val_loss: 0.6969 - val_
accuracy: 0.5000
Epoch 7/10
2/2 [==============================] - 19s 15s/step - loss: 0.7116 - accuracy: 0.5000 - val_loss: 0.7016 - val
_accuracy: 0.5000
Epoch 8/10
2/2 [==============================] - 19s 15s/step - loss: 0.7268 - accuracy: 0.3250 - val_loss: 0.6946 - val
_accuracy: 0.4500
Epoch 9/10
2/2 [==============================] - 21s 6s/step - loss: 0.6702 - accuracy: 0.6250 - val_loss: 0.6963 - val_
accuracy: 0.5000
Epoch 10/10
2/2 [==============================] - 20s 15s/step - loss: 0.6808 - accuracy: 0.5500 - val_loss: 0.6934 - val
_accuracy: 0.5000
```

In [16]:
```
test_loss, test_accuracy = model.evaluate(test)
```

```
1/1 [==============================] - 2s 2s/step - loss: 0.6934 - accuracy: 0.5000
```

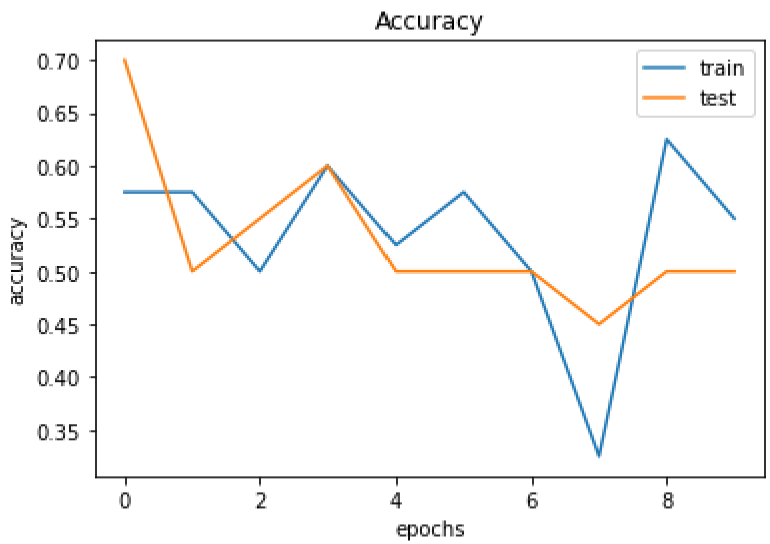
In [17]:
```
test_loss
```

Out[17]:
```
0.6934298276901245
```

In [18]:

```
from matplotlib import pyplot as plt

plt.plot(history.history['accuracy'])
plt.plot(history.history['val_accuracy'])
plt.title('Accuracy')
plt.ylabel('accuracy')
plt.xlabel('epochs')
plt.legend(['train', 'test'], loc = 'upper right')
plt.show()
```



In [19]:
```
history =model.fit(train, validation_data = test, epochs =20)
```

```
Epoch 1/20
2/2 [==============================] - 18s 15s/step - loss: 0.6689 - accuracy: 0.5750 - val_loss: 0.6926 - val
_accuracy: 0.5000
Epoch 2/20
2/2 [==============================] - 18s 6s/step - loss: 0.6385 - accuracy: 0.8250 - val_loss: 0.6976 - val_
accuracy: 0.5000
Epoch 3/20
2/2 [==============================] - 18s 6s/step - loss: 0.6529 - accuracy: 0.6250 - val_loss: 0.6959 - val_
accuracy: 0.5500
Epoch 4/20
2/2 [==============================] - 19s 15s/step - loss: 0.7149 - accuracy: 0.5000 - val_loss: 0.7031 - val
_accuracy: 0.5000
Epoch 5/20
2/2 [==============================] - 20s 16s/step - loss: 0.6441 - accuracy: 0.7750 - val_loss: 0.6957 - val
_accuracy: 0.4000
Epoch 6/20
2/2 [==============================] - 20s 6s/step - loss: 0.6369 - accuracy: 0.7250 - val_loss: 0.6915 - val_
accuracy: 0.4000
Epoch 7/20
2/2 [==============================] - 19s 15s/step - loss: 0.6408 - accuracy: 0.7000 - val_loss: 0.6904 - val
_accuracy: 0.4000
Epoch 8/20
2/2 [==============================] - 21s 16s/step - loss: 0.6529 - accuracy: 0.5750 - val_loss: 0.6893 - val
_accuracy: 0.5000
Epoch 9/20
2/2 [==============================] - 20s 15s/step - loss: 0.6613 - accuracy: 0.5500 - val_loss: 0.6911 - val
_accuracy: 0.4000
Epoch 10/20
2/2 [==============================] - 21s 7s/step - loss: 0.6325 - accuracy: 0.8250 - val_loss: 0.7033 - val_
accuracy: 0.4500
Epoch 11/20
2/2 [==============================] - 19s 5s/step - loss: 0.6214 - accuracy: 0.7000 - val_loss: 0.7009 - val_
accuracy: 0.5000
Epoch 12/20
2/2 [==============================] - 21s 16s/step - loss: 0.6067 - accuracy: 0.7750 - val_loss: 0.6966 - val
_accuracy: 0.4000
Epoch 13/20
2/2 [==============================] - 20s 5s/step - loss: 0.6102 - accuracy: 0.8500 - val_loss: 0.6981 - val_
accuracy: 0.4000
Epoch 14/20
2/2 [==============================] - 19s 5s/step - loss: 0.6150 - accuracy: 0.7500 - val_loss: 0.6986 - val_
```

```
accuracy: 0.5000
Epoch 15/20
2/2 [==============================] - 21s 16s/step - loss: 0.6078 - accuracy: 0.7000 - val_loss: 0.6914 - val
_accuracy: 0.4000
Epoch 16/20
2/2 [==============================] - 21s 6s/step - loss: 0.5902 - accuracy: 0.8250 - val_loss: 0.6961 - val_
accuracy: 0.4500
Epoch 17/20
2/2 [==============================] - 21s 16s/step - loss: 0.6199 - accuracy: 0.6000 - val_loss: 0.6971 - val
_accuracy: 0.4500
Epoch 18/20
2/2 [==============================] - 19s 5s/step - loss: 0.5854 - accuracy: 0.8250 - val_loss: 0.6964 - val_
accuracy: 0.4500
Epoch 19/20
2/2 [==============================] - 21s 16s/step - loss: 0.6247 - accuracy: 0.6750 - val_loss: 0.6986 - val
_accuracy: 0.4500
Epoch 20/20
2/2 [==============================] - 20s 5s/step - loss: 0.5889 - accuracy: 0.7750 - val_loss: 0.7568 - val_
accuracy: 0.5000
```

In [20]:
```
test_loss, test_accuracy = model.evaluate(test)
```

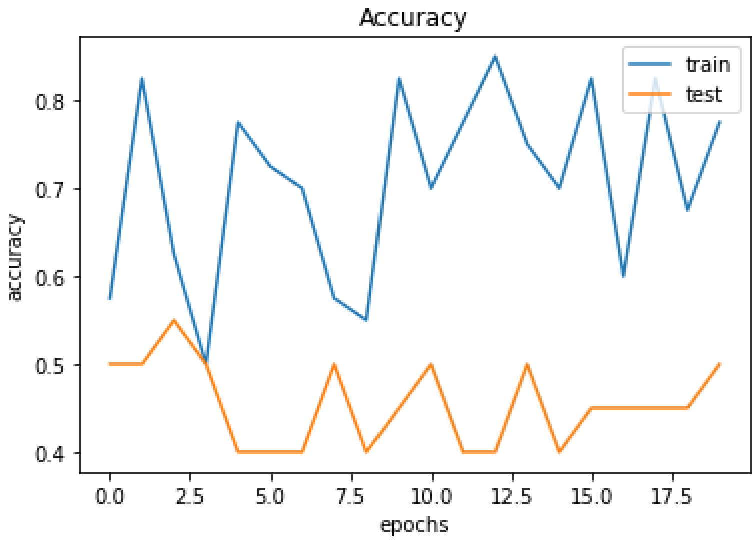
```
1/1 [==============================] - 2s 2s/step - loss: 0.7568 - accuracy: 0.5000
```

In [21]:
```
test_loss
```

Out[21]:
```
0.7567957639694214
```

In [22]:
```
from matplotlib import pyplot as plt

plt.plot(history.history['accuracy'])
plt.plot(history.history['val_accuracy'])
plt.title('Accuracy')
plt.ylabel('accuracy')
plt.xlabel('epochs')
plt.legend(['train', 'test'], loc = 'upper right')
plt.show()
```



In [23]:
```
history =model.fit(train, validation_data = test, epochs =30)
```

```
Epoch 1/30
2/2 [==============================] - 18s 15s/step - loss: 0.5990 - accuracy: 0.7250 - val_loss: 0.6995 - val
_accuracy: 0.4500
Epoch 2/30
2/2 [==============================] - 21s 17s/step - loss: 0.5821 - accuracy: 0.7750 - val_loss: 0.7000 - val
_accuracy: 0.4500
Epoch 3/30
2/2 [==============================] - 21s 7s/step - loss: 0.5718 - accuracy: 0.7750 - val_loss: 0.7104 - val_
accuracy: 0.5500
Epoch 4/30
2/2 [==============================] - 22s 5s/step - loss: 0.5574 - accuracy: 0.8000 - val_loss: 0.7117 - val_
accuracy: 0.4500
Epoch 5/30
2/2 [==============================] - 19s 5s/step - loss: 0.5767 - accuracy: 0.8000 - val_loss: 0.7393 - val_
accuracy: 0.5000
Epoch 6/30
2/2 [==============================] - 20s 16s/step - loss: 0.5712 - accuracy: 0.7000 - val_loss: 0.7005 - val
_accuracy: 0.4500
Epoch 7/30
2/2 [==============================] - 21s 16s/step - loss: 0.5698 - accuracy: 0.8500 - val_loss: 0.7006 - val
_accuracy: 0.4500
Epoch 8/30
2/2 [==============================] - 20s 6s/step - loss: 0.5488 - accuracy: 0.8750 - val_loss: 0.7158 - val_
accuracy: 0.5500
Epoch 9/30
2/2 [==============================] - 19s 5s/step - loss: 0.5460 - accuracy: 0.7750 - val_loss: 0.7016 - val_
accuracy: 0.4500
Epoch 10/30
2/2 [==============================] - 21s 7s/step - loss: 0.5350 - accuracy: 0.8000 - val_loss: 0.7138 - val_
accuracy: 0.4500
Epoch 11/30
2/2 [==============================] - 19s 5s/step - loss: 0.5413 - accuracy: 0.8250 - val_loss: 0.7180 - val_
accuracy: 0.4500
Epoch 12/30
2/2 [==============================] - 19s 5s/step - loss: 0.5607 - accuracy: 0.7750 - val_loss: 0.7669 - val_
accuracy: 0.4500
Epoch 13/30
2/2 [==============================] - 21s 16s/step - loss: 0.6305 - accuracy: 0.6250 - val_loss: 0.7121 - val
_accuracy: 0.4500
Epoch 14/30
2/2 [==============================] - 19s 5s/step - loss: 0.5171 - accuracy: 0.8250 - val_loss: 0.7188 - val_
```

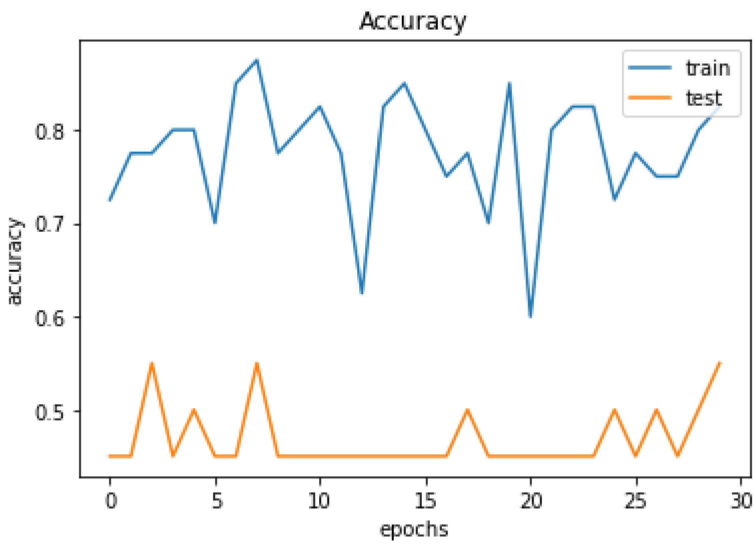
```
accuracy: 0.4500
Epoch 15/30
2/2 [==============================] - 19s 5s/step - loss: 0.5218 - accuracy: 0.8500 - val_loss: 0.7094 - val_
accuracy: 0.4500
Epoch 16/30
2/2 [==============================] - 21s 16s/step - loss: 0.5492 - accuracy: 0.8000 - val_loss: 0.7143 - val
_accuracy: 0.4500
Epoch 17/30
2/2 [==============================] - 21s 16s/step - loss: 0.5405 - accuracy: 0.7500 - val_loss: 0.7173 - val
_accuracy: 0.4500
Epoch 18/30
2/2 [==============================] - 20s 5s/step - loss: 0.5453 - accuracy: 0.7750 - val_loss: 0.8241 - val_
accuracy: 0.5000
Epoch 19/30
2/2 [==============================] - 21s 16s/step - loss: 0.5864 - accuracy: 0.7000 - val_loss: 0.7171 - val
_accuracy: 0.4500
Epoch 20/30
2/2 [==============================] - 21s 16s/step - loss: 0.5045 - accuracy: 0.8500 - val_loss: 0.7170 - val
_accuracy: 0.4500
Epoch 21/30
2/2 [==============================] - 21s 17s/step - loss: 0.6141 - accuracy: 0.6000 - val_loss: 0.7219 - val
_accuracy: 0.4500
Epoch 22/30
2/2 [==============================] - 20s 16s/step - loss: 0.5053 - accuracy: 0.8000 - val_loss: 0.7180 - val
_accuracy: 0.4500
Epoch 23/30
2/2 [==============================] - 21s 16s/step - loss: 0.4982 - accuracy: 0.8250 - val_loss: 0.7219 - val
_accuracy: 0.4500
Epoch 24/30
2/2 [==============================] - 21s 7s/step - loss: 0.4951 - accuracy: 0.8250 - val_loss: 0.7563 - val_
accuracy: 0.4500
Epoch 25/30
2/2 [==============================] - 20s 17s/step - loss: 0.5473 - accuracy: 0.7250 - val_loss: 0.7268 - val
_accuracy: 0.5000
Epoch 26/30
2/2 [==============================] - 20s 16s/step - loss: 0.5031 - accuracy: 0.7750 - val_loss: 0.7226 - val
_accuracy: 0.4500
Epoch 27/30
2/2 [==============================] - 21s 16s/step - loss: 0.5577 - accuracy: 0.7500 - val_loss: 0.7271 - val
_accuracy: 0.5000
Epoch 28/30
2/2 [==============================] - 20s 16s/step - loss: 0.5251 - accuracy: 0.7500 - val_loss: 0.7314 - val
_accuracy: 0.4500
```

```
Epoch 29/30
2/2 [==============================] - 19s 5s/step - loss: 0.4848 - accuracy: 0.8000 - val_loss: 0.7399 - val_
accuracy: 0.5000
Epoch 30/30
2/2 [==============================] - 22s 6s/step - loss: 0.4900 - accuracy: 0.8250 - val_loss: 0.7876 - val_
accuracy: 0.5500
```

In [24]:

```
from matplotlib import pyplot as plt

plt.plot(history.history['accuracy'])
plt.plot(history.history['val_accuracy'])
plt.title('Accuracy')
plt.ylabel('accuracy')
plt.xlabel('epochs')
plt.legend(['train', 'test'], loc = 'upper right')
plt.show()
```



In [25]:

```
test_loss, test_accuracy = model.evaluate(test)
```

```
1/1 [==============================] - 2s 2s/step - loss: 0.7876 - accuracy: 0.5500
```

In [26]:

```
test_loss
```

Out[26]: 0.7875889539718628