```
import zipfile
zip_ref = zipfile.ZipFile('/content/1577957291_deeplearningwithkerasandtensorflow.zip','r')
zip_ref.extractall('/content')
zip_ref.close()
```

```
#import all the require labraries
```

```
import tensorflow as tf
from tensorflow import keras
from keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras import models, layers
```

```
dataflow = ImageDataGenerator(rescale = 1.0/255.0)
```

```
#import training dataset
train = dataflow.flow_from_directory('/content/data/train', class_mode = 'categorical')
```

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

```
#import test data from file
test = dataflow.flow_from_directory('/content/data/test', class_mode='categorical')
```

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

```
#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 layers. This should be fol

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'))
```

```
#For the training step, define the loss function and minimize it
sgd_opt = tf.keras.optimizers.SGD(lr = 0.001)
```

```
/usr/local/lib/python3.8/dist-packages/keras/optimizers/optimizer_v2/gradient_descent.py:108: UserWarning: The `lr` argument is dep
  super(SGD, self).__init__(name, **kwargs)
```

```
#compiling model
history = model.compile( optimizer= sgd_opt,loss = 'binary_crossentropy', metrics = ['accuracy'])
```

```
#model train for 100 epochs
history = model.fit(train, validation_data = test, epochs=100)
```

```
    2/2 [==============================] - 0s 137ms/step - loss: 0.4116 - accuracy: 0.9000 - val_loss: 0.7399 - val_accuracy: 0.4500
    Epoch 84/100
    2/2 [==============================] - 0s 270ms/step - loss: 0.4560 - accuracy: 0.7750 - val_loss: 0.7472 - val_accuracy: 0.4500
    Epoch 85/100
    2/2 [==============================] - 0s 275ms/step - loss: 0.3969 - accuracy: 0.8750 - val_loss: 0.7487 - val_accuracy: 0.4500
    Epoch 86/100
    2/2 [==============================] - 0s 289ms/step - loss: 0.4582 - accuracy: 0.8250 - val_loss: 0.7468 - val_accuracy: 0.4500
    Epoch 87/100
    2/2 [==============================] - 0s 288ms/step - loss: 0.4222 - accuracy: 0.8500 - val_loss: 0.7482 - val_accuracy: 0.4500
    Epoch 88/100
    2/2 [==============================] - 0s 155ms/step - loss: 0.4015 - accuracy: 0.8750 - val_loss: 0.7642 - val_accuracy: 0.5000
    Epoch 89/100
    2/2 [==============================] - 0s 283ms/step - loss: 0.4493 - accuracy: 0.7750 - val_loss: 0.7444 - val_accuracy: 0.5500
    Epoch 90/100
    2/2 [==============================] - 0s 141ms/step - loss: 0.3837 - accuracy: 0.8750 - val_loss: 0.7727 - val_accuracy: 0.5000
    Epoch 91/100
    2/2 [==============================] - 0s 218ms/step - loss: 0.4127 - accuracy: 0.8250 - val_loss: 0.7544 - val_accuracy: 0.5000
    Epoch 92/100
    2/2 [==============================] - 1s 556ms/step - loss: 0.4165 - accuracy: 0.8000 - val_loss: 0.7462 - val_accuracy: 0.4000
    Epoch 93/100
    2/2 [==============================] - 1s 215ms/step - loss: 0.3738 - accuracy: 0.9000 - val_loss: 0.7696 - val_accuracy: 0.5000
    Epoch 94/100
    2/2 [==============================] - 1s 353ms/step - loss: 0.4314 - accuracy: 0.8250 - val_loss: 0.8249 - val_accuracy: 0.5000
    Epoch 95/100
    2/2 [==============================] - 1s 240ms/step - loss: 0.4132 - accuracy: 0.8250 - val_loss: 0.9277 - val_accuracy: 0.5000
    Epoch 96/100
    2/2 [==============================] - 1s 235ms/step - loss: 0.5091 - accuracy: 0.7000 - val_loss: 0.7595 - val_accuracy: 0.4500
    Epoch 97/100
    2/2 [==============================] - 1s 252ms/step - loss: 0.3544 - accuracy: 0.9250 - val_loss: 0.7608 - val_accuracy: 0.4500
    Epoch 98/100
    2/2 [==============================] - 1s 224ms/step - loss: 0.3740 - accuracy: 0.8500 - val_loss: 0.7641 - val_accuracy: 0.4500
    Epoch 99/100
    2/2 [==============================] - 1s 232ms/step - loss: 0.3689 - accuracy: 0.8750 - val_loss: 0.8670 - val_accuracy: 0.6000
    Epoch 100/100
    2/2 [==============================] - 1s 486ms/step - loss: 0.4161 - accuracy: 0.8000 - val_loss: 0.7753 - val_accuracy: 0.4500
```

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

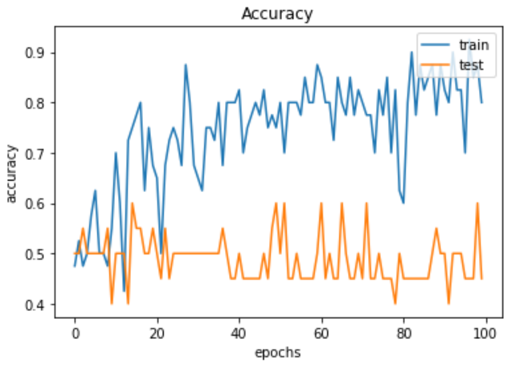
```
    1/1 [==============================] - 0s 322ms/step - loss: 0.7753 - accuracy: 0.4500
```

```
test_loss
```

```
    0.7753384709358215
```

```
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()
```



```
history =model.fit(train, validation_data = test, epochs =200)
```

```
Epoch 177/200
2/2 [==============================] - 0s 279ms/step - loss: 0.0745 - accuracy: 1.0000 - val_loss: 1.0434 - val_accuracy: 0.6000
Epoch 178/200
2/2 [==============================] - 0s 144ms/step - loss: 0.0581 - accuracy: 1.0000 - val_loss: 1.0551 - val_accuracy: 0.6000
Epoch 179/200
2/2 [==============================] - 0s 284ms/step - loss: 0.1089 - accuracy: 0.9750 - val_loss: 1.0479 - val_accuracy: 0.6000
Epoch 180/200
2/2 [==============================] - 0s 143ms/step - loss: 0.0720 - accuracy: 1.0000 - val_loss: 1.1717 - val_accuracy: 0.5500
Epoch 181/200
2/2 [==============================] - 0s 144ms/step - loss: 0.1107 - accuracy: 0.9750 - val_loss: 1.0779 - val_accuracy: 0.6000
Epoch 182/200
2/2 [==============================] - 0s 138ms/step - loss: 0.0684 - accuracy: 0.9750 - val_loss: 1.0997 - val_accuracy: 0.5500
Epoch 183/200
2/2 [==============================] - 0s 137ms/step - loss: 0.0637 - accuracy: 1.0000 - val_loss: 1.0718 - val_accuracy: 0.6000
Epoch 184/200
2/2 [==============================] - 0s 291ms/step - loss: 0.0620 - accuracy: 1.0000 - val_loss: 1.0816 - val_accuracy: 0.5000
Epoch 185/200
2/2 [==============================] - 0s 280ms/step - loss: 0.0668 - accuracy: 1.0000 - val_loss: 1.0838 - val_accuracy: 0.6000
Epoch 186/200
2/2 [==============================] - 0s 282ms/step - loss: 0.1170 - accuracy: 1.0000 - val_loss: 1.0973 - val_accuracy: 0.6000
Epoch 187/200
2/2 [==============================] - 0s 153ms/step - loss: 0.0741 - accuracy: 0.9750 - val_loss: 1.0658 - val_accuracy: 0.6000
Epoch 188/200
2/2 [==============================] - 0s 283ms/step - loss: 0.0599 - accuracy: 1.0000 - val_loss: 1.0721 - val_accuracy: 0.5500
Epoch 189/200
2/2 [==============================] - 0s 287ms/step - loss: 0.0582 - accuracy: 1.0000 - val_loss: 1.0648 - val_accuracy: 0.6000
Epoch 190/200
2/2 [==============================] - 0s 145ms/step - loss: 0.0809 - accuracy: 0.9750 - val_loss: 1.2971 - val_accuracy: 0.5000
Epoch 191/200
2/2 [==============================] - 0s 294ms/step - loss: 0.2180 - accuracy: 0.8750 - val_loss: 1.2689 - val_accuracy: 0.5000
Epoch 192/200
2/2 [==============================] - 0s 151ms/step - loss: 0.1113 - accuracy: 1.0000 - val_loss: 1.0833 - val_accuracy: 0.6000
Epoch 193/200
2/2 [==============================] - 0s 144ms/step - loss: 0.0551 - accuracy: 1.0000 - val_loss: 1.0771 - val_accuracy: 0.6000
Epoch 194/200
2/2 [==============================] - 0s 288ms/step - loss: 0.0540 - accuracy: 1.0000 - val_loss: 1.0904 - val_accuracy: 0.5000
Epoch 195/200
2/2 [==============================] - 0s 146ms/step - loss: 0.0435 - accuracy: 1.0000 - val_loss: 1.0815 - val_accuracy: 0.6000
Epoch 196/200
2/2 [==============================] - 0s 279ms/step - loss: 0.0715 - accuracy: 1.0000 - val_loss: 1.1069 - val_accuracy: 0.5000
Epoch 197/200
2/2 [==============================] - 0s 141ms/step - loss: 0.0418 - accuracy: 1.0000 - val_loss: 1.1016 - val_accuracy: 0.6000
Epoch 198/200
2/2 [==============================] - 0s 145ms/step - loss: 0.0482 - accuracy: 1.0000 - val_loss: 1.1182 - val_accuracy: 0.5500
Epoch 199/200
2/2 [==============================] - 0s 278ms/step - loss: 0.0499 - accuracy: 1.0000 - val_loss: 1.0806 - val_accuracy: 0.6000
Epoch 200/200
2/2 [==============================] - 0s 141ms/step - loss: 0.0425 - accuracy: 1.0000 - val_loss: 1.0825 - val_accuracy: 0.6000
```

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

```
1/1 [==============================] - 0s 126ms/step - loss: 1.0825 - accuracy: 0.6000
```

```
test_loss
```

```
1.0825061798095703
```

```
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()
```

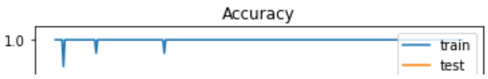
Accuracy

```
history =model.fit(train, validation_data = test, epochs =300)
```

```
Epoch 272/300
2/2 [==============================] - 0s 291ms/step - loss: 0.0074 - accuracy: 1.0000 - val_loss: 1.5042 - val_accuracy: 0.6000
Epoch 273/300
2/2 [==============================] - 0s 137ms/step - loss: 0.0080 - accuracy: 1.0000 - val_loss: 1.5094 - val_accuracy: 0.6000
Epoch 274/300
2/2 [==============================] - 0s 147ms/step - loss: 0.0081 - accuracy: 1.0000 - val_loss: 1.5004 - val_accuracy: 0.6000
Epoch 275/300
2/2 [==============================] - 0s 294ms/step - loss: 0.0115 - accuracy: 1.0000 - val_loss: 1.5056 - val_accuracy: 0.6000
Epoch 276/300
2/2 [==============================] - 0s 282ms/step - loss: 0.0072 - accuracy: 1.0000 - val_loss: 1.5102 - val_accuracy: 0.6000
Epoch 277/300
2/2 [==============================] - 0s 295ms/step - loss: 0.0101 - accuracy: 1.0000 - val_loss: 1.5095 - val_accuracy: 0.6000
Epoch 278/300
2/2 [==============================] - 0s 142ms/step - loss: 0.0082 - accuracy: 1.0000 - val_loss: 1.5068 - val_accuracy: 0.6000
Epoch 279/300
2/2 [==============================] - 0s 283ms/step - loss: 0.0060 - accuracy: 1.0000 - val_loss: 1.5047 - val_accuracy: 0.6000
Epoch 280/300
2/2 [==============================] - 0s 141ms/step - loss: 0.0068 - accuracy: 1.0000 - val_loss: 1.5051 - val_accuracy: 0.6000
Epoch 281/300
2/2 [==============================] - 0s 144ms/step - loss: 0.0074 - accuracy: 1.0000 - val_loss: 1.5095 - val_accuracy: 0.6000
Epoch 282/300
2/2 [==============================] - 0s 281ms/step - loss: 0.0074 - accuracy: 1.0000 - val_loss: 1.5099 - val_accuracy: 0.6000
Epoch 283/300
2/2 [==============================] - 0s 282ms/step - loss: 0.0089 - accuracy: 1.0000 - val_loss: 1.5217 - val_accuracy: 0.6000
Epoch 284/300
2/2 [==============================] - 0s 288ms/step - loss: 0.0096 - accuracy: 1.0000 - val_loss: 1.5168 - val_accuracy: 0.6000
Epoch 285/300
2/2 [==============================] - 0s 135ms/step - loss: 0.0088 - accuracy: 1.0000 - val_loss: 1.5158 - val_accuracy: 0.6000
Epoch 286/300
2/2 [==============================] - 0s 292ms/step - loss: 0.0074 - accuracy: 1.0000 - val_loss: 1.5126 - val_accuracy: 0.6000
Epoch 287/300
2/2 [==============================] - 0s 286ms/step - loss: 0.0099 - accuracy: 1.0000 - val_loss: 1.5181 - val_accuracy: 0.6000
Epoch 288/300
2/2 [==============================] - 0s 155ms/step - loss: 0.0119 - accuracy: 1.0000 - val_loss: 1.5383 - val_accuracy: 0.5500
Epoch 289/300
2/2 [==============================] - 0s 140ms/step - loss: 0.0102 - accuracy: 1.0000 - val_loss: 1.5397 - val_accuracy: 0.5500
Epoch 290/300
2/2 [==============================] - 0s 154ms/step - loss: 0.0069 - accuracy: 1.0000 - val_loss: 1.5360 - val_accuracy: 0.6000
Epoch 291/300
2/2 [==============================] - 0s 153ms/step - loss: 0.0091 - accuracy: 1.0000 - val_loss: 1.5270 - val_accuracy: 0.6000
Epoch 292/300
2/2 [==============================] - 0s 142ms/step - loss: 0.0079 - accuracy: 1.0000 - val_loss: 1.5315 - val_accuracy: 0.6000
Epoch 293/300
2/2 [==============================] - 0s 289ms/step - loss: 0.0113 - accuracy: 1.0000 - val_loss: 1.5552 - val_accuracy: 0.5500
Epoch 294/300
2/2 [==============================] - 0s 144ms/step - loss: 0.0079 - accuracy: 1.0000 - val_loss: 1.5483 - val_accuracy: 0.5500
Epoch 295/300
2/2 [==============================] - 0s 300ms/step - loss: 0.0075 - accuracy: 1.0000 - val_loss: 1.5378 - val_accuracy: 0.6000
Epoch 296/300
2/2 [==============================] - 0s 283ms/step - loss: 0.0063 - accuracy: 1.0000 - val_loss: 1.5422 - val_accuracy: 0.6000
Epoch 297/300
2/2 [==============================] - 0s 303ms/step - loss: 0.0078 - accuracy: 1.0000 - val_loss: 1.5421 - val_accuracy: 0.6000
Epoch 298/300
2/2 [==============================] - 0s 138ms/step - loss: 0.0096 - accuracy: 1.0000 - val_loss: 1.5446 - val_accuracy: 0.6000
Epoch 299/300
2/2 [==============================] - 0s 286ms/step - loss: 0.0079 - accuracy: 1.0000 - val_loss: 1.5477 - val_accuracy: 0.6000
Epoch 300/300
2/2 [==============================] - 0s 138ms/step - loss: 0.0090 - accuracy: 1.0000 - val_loss: 1.5494 - val_accuracy: 0.6000
```

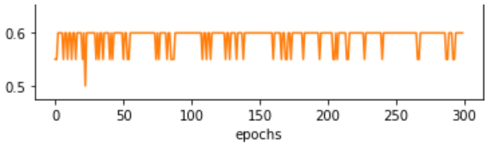
```
from matplotlib import pyplot as plt

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()
```



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

```
    1/1 [==============================] - 0s 123ms/step - loss: 1.5494 - accuracy: 0.6000
    1.5494447946548462
```



Colab paid products - Cancel contracts here

✓ 0s completed at 10:59 AM