Predictions\_on\_Dataset\_and\_Plotting\_the\_Accuracy.ipynb

515 lines (515 sloc) 46.9 KB

In [1]:

**import** tensorflow **as** tf

**import** numpy **as** np

**from** tensorflow **import** keras

**import** os

**import** cv2

**from** tensorflow.keras.preprocessing.image **import** ImageDataGenerator

**from** tensorflow.keras.preprocessing **import** image

**import** matplotlib.pyplot **as** plt

# Making Separate Dataset for Training and Testing

In [2]:

train **=** ImageDataGenerator(rescale**=**1**/**255) test **=** ImageDataGenerator(rescale**=**1**/**255)

train\_dataset **=** train**.**flow\_from\_directory("/content/drive/MyDrive/Dataset/tra

target\_size**=**(150,150), batch\_size **=** 32,

class\_mode **=** 'binary')

test\_dataset **=** test**.**flow\_from\_directory("/content/drive/MyDrive/Dataset/test\_

target\_size**=**(150,150), batch\_size **=**32,

class\_mode **=** 'binary')

In [3]:

test\_dataset**.**class\_indices

Found 436 images belonging to 2 classes. Found 121 images belonging to 2 classes.

Out[3]:

In [6]:

model **=** keras**.**Sequential()

model**.**add(keras**.**layers**.**Conv2D(32,(3,3),activation**=**'relu',input\_shape**=**(150,150 model**.**add(keras**.**layers**.**MaxPool2D(2,2))

model**.**add(keras**.**layers**.**Conv2D(64,(3,3),activation**=**'relu')) model**.**add(keras**.**layers**.**MaxPool2D(2,2))

model**.**add(keras**.**layers**.**Conv2D(128,(3,3),activation**=**'relu'))

model**.**add(keras**.**layers**.**MaxPool2D(2,2))

model**.**add(keras**.**layers**.**Conv2D(128,(3,3),activation**=**'relu')) model**.**add(keras**.**layers**.**MaxPool2D(2,2))

model**.**add(keras**.**layers**.**Flatten())

model**.**add(keras**.**layers**.**Dense(512,activation**=**'relu')) model**.**add(keras**.**layers**.**Dense(1,activation**=**'sigmoid'))

{'forest': 0, 'with fire': 1}

# Model Building

Compiling the model

In [7]:

model**.**compile(optimizer**=**'adam',loss**=**'binary\_crossentropy',metrics**=**['accuracy'



# Fitting the Model

In [8]:

r **=** model**.**fit(train\_dataset,

epochs **=** 10,

validation\_data **=** test\_dataset)

|  |  |  |
| --- | --- | --- |
| Epoch 1/10 |  | |
| 14/14 [==============================] - 150s 11s/step  acy: 0.7339 - val\_loss: 0.2174 - val\_accuracy: 0.8843 Epoch 2/10 | - loss: 0.4300 | - accur |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 14/14 [==============================] - 23s | 2s/step | - loss: | 0.1994 | - accurac |
| y: 0.9037 - val\_loss: 0.0567 - val\_accuracy: Epoch 3/10  14/14 [==============================] - 23s | 0.9835  2s/step | - loss: | 0.2111 | - accurac |
| y: 0.9014 - val\_loss: 0.0940 - val\_accuracy: | 0.9835 |  |  |  |
| Epoch 4/10  14/14 [==============================] - 23s | 2s/step | - loss: | 0.1536 | - accurac |
| y: 0.9335 - val\_loss: 0.0322 - val\_accuracy:  Epoch 5/10 | 1.0000 |  |  |  |
| 14/14 [==============================] - 23s  y: 0.9564 - val\_loss: 0.0130 - val\_accuracy: Epoch 6/10  14/14 [==============================] - 23s | 2s/step 1.0000  2s/step | * loss: * loss: | 0.1192  0.1265 | * accurac * accurac |
| y: 0.9564 - val\_loss: 0.0633 - val\_accuracy: Epoch 7/10  14/14 [==============================] - 23s | 0.9917  2s/step | - loss: | 0.0969 | - accurac |
| y: 0.9725 - val\_loss: 0.0238 - val\_accuracy: | 0.9917 |  |  |  |
| Epoch 8/10  14/14 [==============================] - 23s | 2s/step | - loss: | 0.1475 | - accurac |
| y: 0.9312 - val\_loss: 0.0352 - val\_accuracy:  Epoch 9/10 | 1.0000 |  |  |  |
| 14/14 [==============================] - 23s  y: 0.9610 - val\_loss: 0.0230 - val\_accuracy: Epoch 10/10  14/14 [==============================] - 23s | 2s/step 1.0000  2s/step | * loss: * loss: | 0.0955  0.0662 | * accurac * accurac |
| y: 0.9748 - val\_loss: 0.0147 - val\_accuracy: | 1.0000 |  |  |  |
| Predictions on Dataset |  |  |  |  |

In [10]:

predictions **=** model**.**predict(test\_dataset) predictions **=** np**.**round(predictions)

4/4 [==============================] - 4s 1s/step

In [11]:

predictions

Out[11]:

array([[1.],

[0.],

[0.],

[1.],

[0.],

[1.],

[0.],

[0.],

[0.],

[1.],

[1.],

[1.],

[0.],

[0.],

[0.],

[1.],

[0.],

[0.],

[1.],

[1.],

[1.],

[0.],

[0.],

[0.],

[1.],

[1.],

[1.],

[0.],

[0.],

[0.],

[1.],

[1.],

[0.],

[1.],

[1.],

[1.],

[0.],

[0.],

[1.],

[1.],

[0.],

[0.],

[1.],

[1.],

[0.],

[1.],

[1.],

[0.],

[0.],

[0.],

[0.],

[0.],

[1.],

[1.],

[0.],

[0.],

[0.],

[0.],

[1.],

[0.],

[1.],

[1.],

[0.],

[0.],

[0.],

[0.],

[0.],

[0.],

[1.],

[1.],

[1.],

[0.],

[0.],

[1.],

[0.],

[0.],

[0.],

[1.],

[0.],

[0.],

[0.],

[0.],

[1.],

[0.],

[0.],

[0.],

[0.],

[1.],

[0.],

[0.],

[1.],

[1.],

[0.],

[0.],

[0.],

[0.],

[1.],

[1.],

[0.],

[1.],

[0.],

[0.],

[0.],

[0.],

[1.],

[0.],

[0.],

[0.],

[0.],

[1.],

[0.],

[1.],

[0.],

[0.],

[1.],

[1.],

[1.],

[1.],

[0.],

[0.],

[1.]], dtype=float32)

In [12]:

print(len(predictions))

121

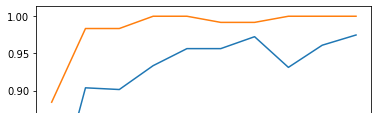
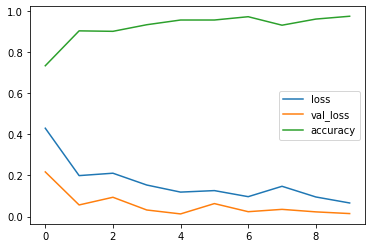
# Plotting Loss per Iteration

In [13]:

**import** matplotlib.pyplot **as** plt

plt**.**plot(r**.**history['loss'], label**=**'loss')

plt**.**plot(r**.**history['val\_loss'], label**=**'val\_loss')



plt**.**plot(r**.**history['accuracy'], label**=**'accuracy')

plt**.**legend()

Out[13]:

Plotting accuracy per Iteration

In [14]:

Out[14]:

plt**.**plot(r**.**history['accuracy'], label**=**'acc')

plt**.**plot(r**.**history['val\_accuracy'], label**=**'val\_acc') plt**.**legend()