Ex No: 3 **Date: 09-08-2024**

BUILD A CONVOLUTIONAL NEURAL NETWORK

AIM:

To build a simple convolutional neural network with Keras/TensorFlow.

PROCEDURE:

1. Download and load the dataset.

2. Perform analysis and preprocessing of the dataset.

3. Build a convolutional neural network model using Keras/TensorFlow.

4. Compile and fit the model.

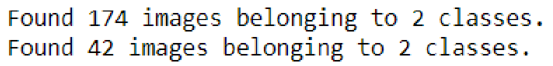
5. Perform prediction with the test dataset.

6. Calculate performance metrics.

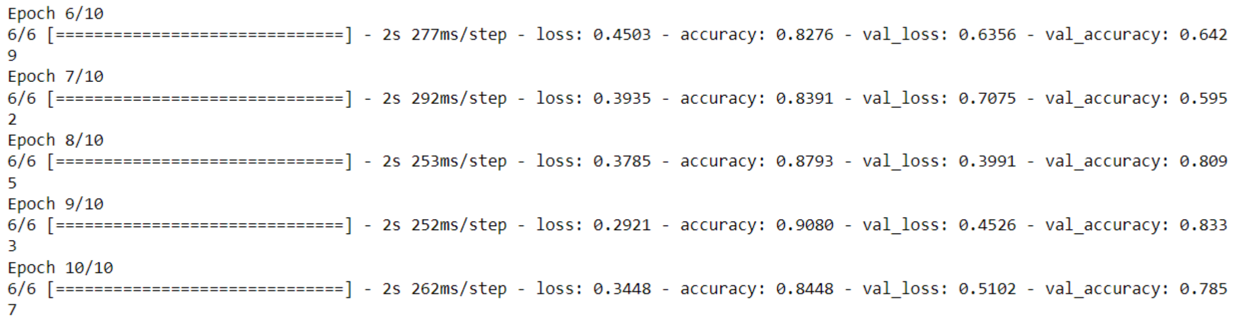
PROGRAM:

```
from tensorflow.keras.layers import Dense, Flatten, Conv2D, MaxPooling2D
from tensorflow.keras.models import Model
from tensorflow.keras.applications.vgg19 import VGG19
from tensorflow.keras.applications.resnet50 import preprocess_input
from tensorflow.keras.preprocessing import image
from tensorflow.keras.preprocessing.image import ImageDataGenerator,load_img
from tensorflow.keras.models import Sequential
import numpy as np
import matplotlib.pyplot as plt
model=Sequential()
model.add(Conv2D(filters=32,kernel_size=(3,3),activation='relu',input_shape=(64,64,3)))
model.add(MaxPooling2D(pool_size=(2,2)))
model.add(Conv2D(filters=64,kernel_size=(3,3),activation ='relu'))
model.add(MaxPooling2D(pool_size=(2,2)))
model.add(Conv2D(filters=128,kernel_size=(3,3),activation='relu'))
model.add(MaxPooling2D(pool_size=(2,2)))
model.add(Flatten())
model.add(Dense(units=128,activation='relu'))
model.add(Dense(units=1,activation='sigmoid'))
model.compile(loss='binary_crossentropy',optimizer='adam',metrics=['accuracy'])
dataset_dir = 'cell_images/cell_images/'
train_datagen = ImageDataGenerator(rescale=1.0/255, shear_range=0.2, zoom_range=0.2,
horizontal_flip=True, validation_split=0.2  # 20% of data for validation)
test_datagen = ImageDataGenerator(rescale=1.0/255)
training_set = train_datagen.flow_from_directory(dataset_dir, target_size=(64, 64),
batch_size=32, class_mode='binary',  # Use 'binary' for binary classification subset='training')
validation_set = train_datagen.flow_from_directory(dataset_dir, target_size=(64, 64),
batch_size=32, class_mode='binary',  subset='validation')
r=model.fit(training_set,validation_data=validation_set,epochs=10,
steps_per_epoch=len(training_set), validation_steps=len(validation_set) )
plt.plot(r.history['loss'], label='train loss')
plt.plot(r.history['val_loss'], label='val loss')
plt.legend()
plt.show()
plt.savefig('LossVal_loss')
plt.plot(r.history['accuracy'], label='train acc')
plt.plot(r.history['val_accuracy'], label='val acc')
plt.legend()
plt.show()
plt.savefig('AccVal_acc')
```

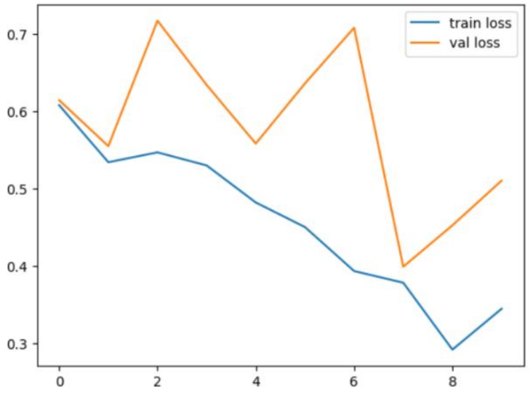
OUTPUT:



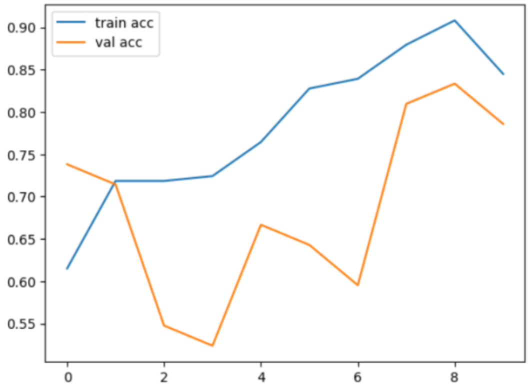
```
Found 174 images belonging to 2 classes.
Found 42 images belonging to 2 classes.
```



```
Epoch 6/10
6/6 [==============================] - 2s 277ms/step - loss: 0.4503 - accuracy: 0.8276 - val_loss: 0.6356 - val_accuracy: 0.642
9
Epoch 7/10
6/6 [==============================] - 2s 292ms/step - loss: 0.3935 - accuracy: 0.8391 - val_loss: 0.7075 - val_accuracy: 0.595
2
Epoch 8/10
6/6 [==============================] - 2s 253ms/step - loss: 0.3785 - accuracy: 0.8793 - val_loss: 0.3991 - val_accuracy: 0.809
5
Epoch 9/10
6/6 [==============================] - 2s 252ms/step - loss: 0.2921 - accuracy: 0.9080 - val_loss: 0.4526 - val_accuracy: 0.833
3
Epoch 10/10
6/6 [==============================] - 2s 262ms/step - loss: 0.3448 - accuracy: 0.8448 - val_loss: 0.5102 - val_accuracy: 0.785
7
```



```
<Figure size 640x480 with 0 Axes>
```



```
<Figure size 640x480 with 0 Axes>
```

RESULT:

Thus, a simple convolutional neural network with Keras/TensorFlow was successfully implemented.