Rachid LAMJOUN
AI & ML Master Program

Pet Classification Model Using CNN

(AI Capstone Project: Project3 Pet Classification Model Using CNN)

Writeup

We have the subject to build a CNN model that classifies the given pet images correctly into dog and cat images. The 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

We must use the activation function **softmax** on the output of the dropout layers and respect the two points for training and evaluation:

- For the training step, define the loss function and minimize it
- For the evaluation step, calculate the accuracy

To achieve our goal, we have opted for the following choices:

- We presented the data by the object **ImageDataGenerator** of **tensorflow.keras.preprocessing.image**.
- we have created the **get_model()** function which generates a model which respects the choice of layers above,
- the function **fit_model()** to train the model,
- the function **eval()** for evaluating the loss and the accuracy of model,
- and in the end, we created **plot_loss_acc()** function to show the evolution of loss and accuracy metrics according to the epochs carried out.

For iterations 100, 200 and 300 we got the following results:

```
#-------------------
# model_100: Evaluate on Validation data
#-------------------
print('loss and accuracy on the Evaluation Data:')
print('--------')
eval(model_100,validation_generator)
print('--------')
```

```
loss and accuracy on the Evaluation Data:
--------
loss= 576.29 %
accuracy= 60.0 %
--------
```

```
#-------------------
# model_200: Evaluate on Validation data
#-------------------
print('loss and accuracy on the Evaluation Data:')
print('--------')
eval(model_200,validation_generator)
print('--------')
```

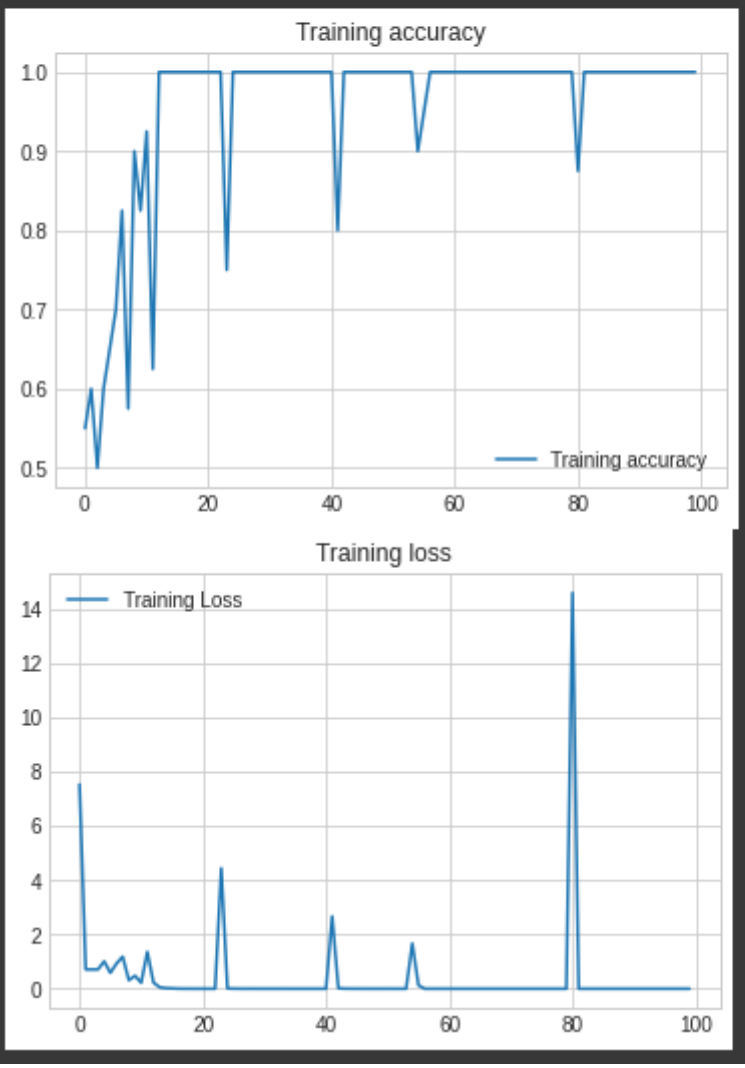
```
loss and accuracy on the Evaluation Data:
--------
loss= 1258.25 %
accuracy= 50.0 %
--------
```

```
#-------------------
# model_300: Evaluate on Validation data
#-------------------
print('loss and accuracy on the Evaluation Data:')
print('--------')
eval(model_300,validation_generator)
print('--------')
```

```
loss and accuracy on the Evaluation Data:
--------
loss= 1699.82 %
accuracy= 55.0 %
--------
```

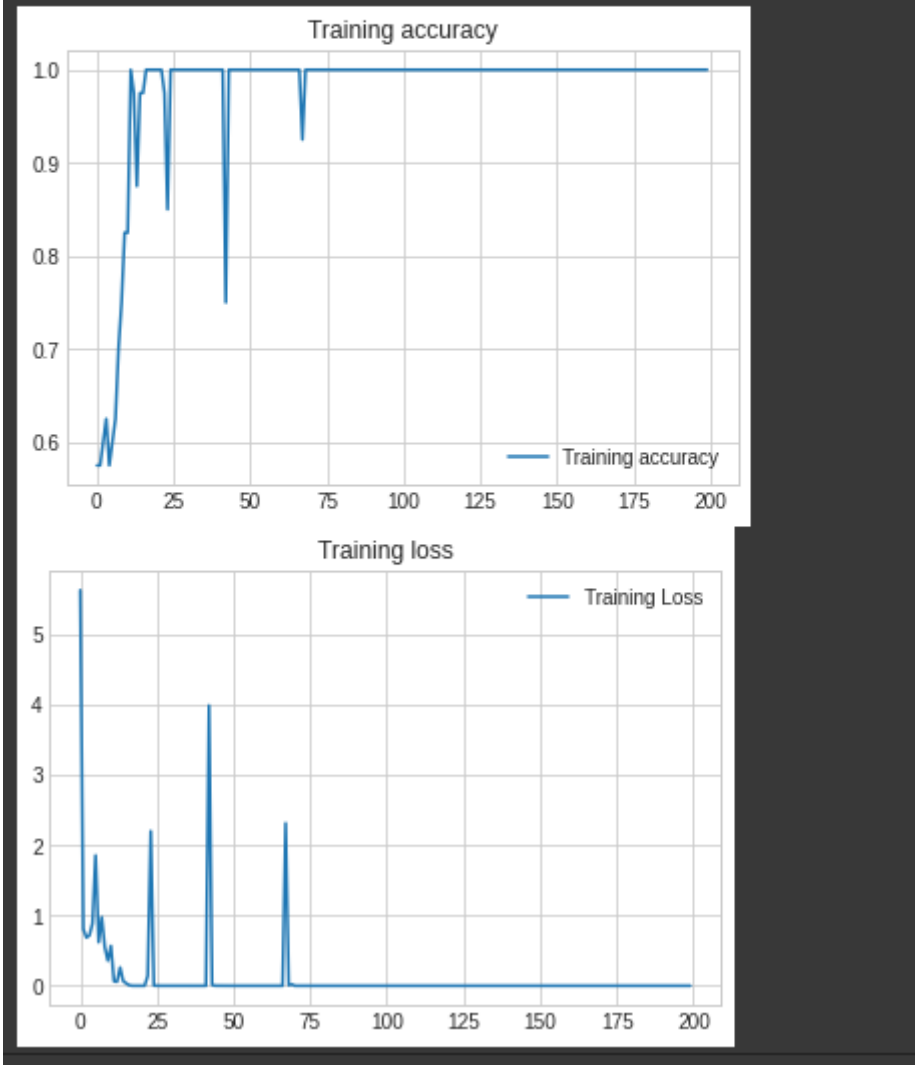
Tracking accuracy and loss for iteration 100

```
#-----------
epochs=100
#-----------
#
model_100,history_100=fit_model(get_model(),epochs,0.001)

plot_loss_acc(history_100)
```



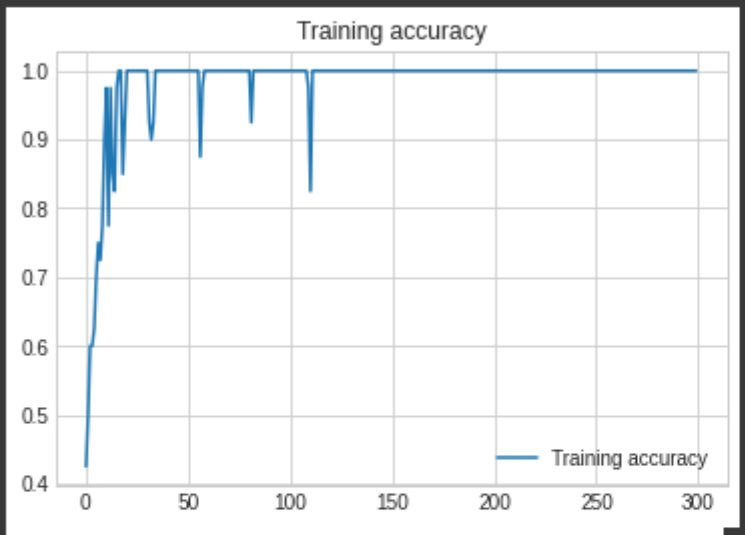
Tracking accuracy and loss for iteration 200

```
#-----------
epochs=200
#-----------
#
model_200,history_200=fit_model(get_model(),epochs,0.001)

plot_loss_acc(history_200)
```



Tracking accuracy and loss for iteration 300

```
#-----------
epochs=300
#-----------
#
model_300,history_300=fit_model(get_model(),epochs,0.001)

plot_loss_acc(history_300)
```



DETAILS

ImageDataGenerator Objects

```
import numpy as np
import tensorflow as tf
import urllib
import zipfile
import matplotlib.pyplot as plt
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Input, Dense, Conv2D, Dropout, Flatten, MaxPooling2D

local_zip = '/content/data_pet_classification.zip'
zip_ref = zipfile.ZipFile(local_zip, 'r')
zip_ref.extractall('content/pet_classification/')
zip_ref.close()
#
IMAGE_SHAPE=300
img_train_folder='content/pet_classification/data/train/'
img_test_folder='content/pet_classification/data/test/'
#
train_datagen = ImageDataGenerator(rescale=1/255)

validation_datagen = ImageDataGenerator(rescale=1/255)


train_generator = train_datagen.flow_from_directory(
                            img_train_folder,
                            target_size=(300, 300),
                            batch_size=4,
                            class_mode='binary')

validation_generator = validation_datagen.flow_from_directory(
                            img_test_folder,
                            target_size=(300, 300),
                            batch_size=4,
                            class_mode='binary')

inputShape=(IMAGE_SHAPE, IMAGE_SHAPE, 3)
fc_size=32
```

get_model() function:

```
#-------------------------
#--- Generate le CNN model
#-------------------------
def get_model():
 l_model=Sequential()
 l_model.add(Input(inputShape))
 l_model.add(Conv2D(32, kernel_size=(5,5),activation='relu'))
 l_model.add(MaxPooling2D(pool_size=(2,2)))
 l_model.add(Conv2D(64, kernel_size=(5,5),activation='relu'))
 l_model.add(MaxPooling2D(pool_size=(2,2)))
 l_model.add(Dense(fc_size,activation='relu'))
 l_model.add(Flatten())
 l_model.add(Dropout(0.4))
 l_model.add(Dense(2,activation='softmax'))

 #l_model.summary()

 return l_model
```

fit_model() function:

```
#-------------------------
#--- For traning model
#-------------------------
def fit_model(i_model,i_num_epoch,i_lr,i_verbose=0):
 l_model=i_model
 l_model.compile(loss='sparse_categorical_crossentropy',
             optimizer=tf.keras.optimizers.RMSprop(learning_rate=i_lr),
             metrics=['accuracy'])

 l_history= l_model.fit(
                train_generator,
                epochs=i_num_epoch,
                verbose=i_verbose
                )
 return l_model, l_history
```

eval() function:

```
#-------------------------
#--- For loss and accuracy evaluating
#-------------------------
def eval(i_model,i_validation_generator):
 l_scores = i_model.evaluate(i_validation_generator,verbose=0)
 print('loss=',np.round(l_scores[0]*100,2),'%')
 print('accuracy=',np.round(l_scores[1]*100,2),'%')
```

plot_loss_acc() function:

```
#-------------------------
#--- To plot loss and accuracy vs epochs
#-------------------------
def plot_loss_acc(i_history):

 acc = i_history.history['accuracy']
 loss = i_history.history['loss']

 epochs = range(len(acc))

 plt.style.use('seaborn-whitegrid')
 #plt.figure(figsize=(6, 4))
 plt.plot(epochs, acc, label='Training accuracy')
 plt.title('Training accuracy')
 plt.legend()
 #
 plt.figure()
 #plt.plot(epochs, loss, 'bo', label='Training Loss')
 plt.plot(epochs, loss, label='Training Loss')
 plt.title('Training loss')
 plt.legend()
```