Name - Yash Gandhi TE IT Batch A 2017140014

Title : Image Classification using Convolutional Neural Network

Theory :

A convolutional neural network consists of an input and an output layer, as well as multiple hidden layers. The hidden layers of a CNN typically consist of a series of convolutional layers that convolve with a multiplication or other dot product. The activation function is commonly a RELU layer, and is subsequently followed by additional convolutions such as pooling layers, fully connected layers and normalization layers, referred to as hidden layers because their inputs and outputs are masked by the activation function and final convolution.

Implementation :

Steps :

1. Import the image dataset
2. Initialize the hyper Paramets i.e Epoch,Batch Size,Image width and height etc required
3. Data Scaling to normalize the image size
4. Split the dataset into train and test
5. Using Sequential Model add all the hidden layers of the CNN and build the model using keras
6. Train the Model using training data
7. Find the accuracy and loss curve
8. Predict the test data and print the result

Code :

```
# In[1]:


get_ipython().magic(u'matplotlib inline')
import matplotlib.pyplot as plt
import matplotlib.image as mpimg
from keras.preprocessing.image import ImageDataGenerator
from keras.models import Sequential, Model
from keras.optimizers import RMSprop
from keras.layers import Activation, Dropout, Flatten, Dense, GlobalMaxPooling2D, Conv2D,
MaxPooling2D
from keras.callbacks import CSVLogger
from livelossplot.keras import PlotLossesCallback
```

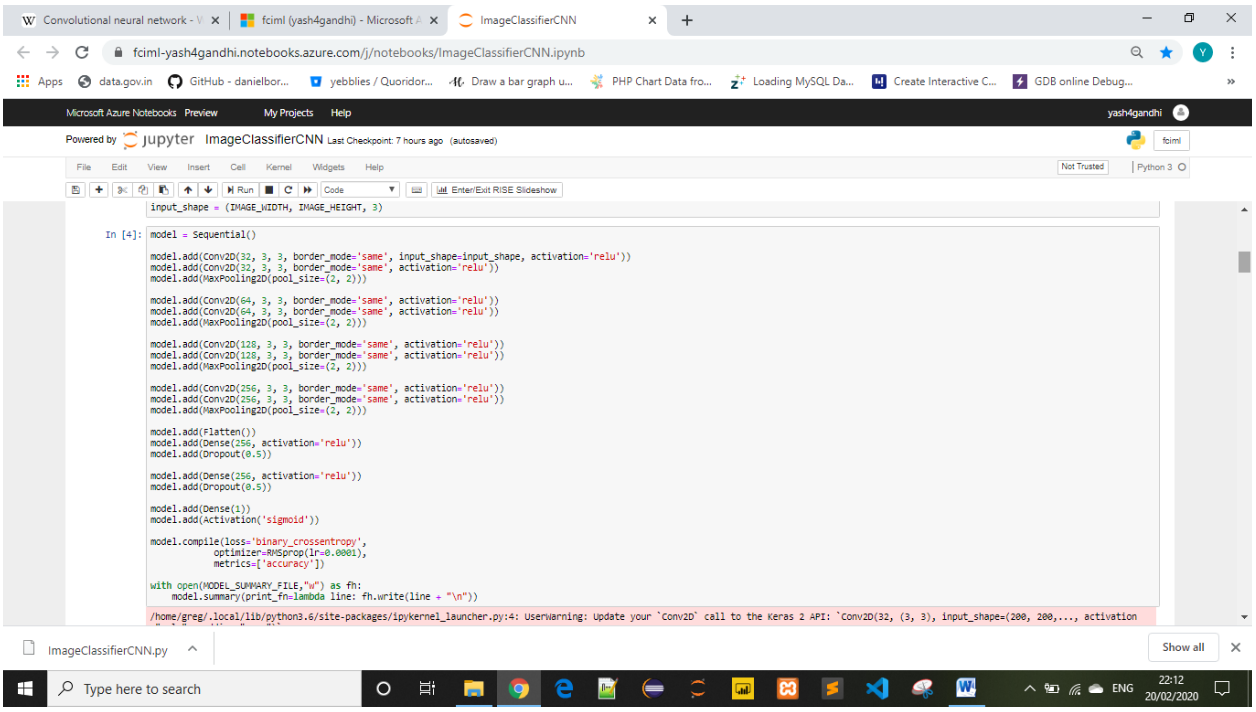
```
import efficientnet.keras as efn


# In[2]:




# Data
path = "/home/greg/datasets/cats_and_dogs/"
training_data_dir = path + "training" # 10 000 * 2
validation_data_dir = path + "validation" # 2 500 * 2
test_data_dir = path + "test" # 12 500


# In[3]:


# Hyperparams
IMAGE_SIZE = 200
IMAGE_WIDTH, IMAGE_HEIGHT = IMAGE_SIZE, IMAGE_SIZE
EPOCHS = 20
BATCH_SIZE = 32
TEST_SIZE = 30

input_shape = (IMAGE_WIDTH, IMAGE_HEIGHT, 3)


# In[4]:


model = Sequential()

model.add(Conv2D(32, 3, 3, border_mode='same', input_shape=input_shape, activation='relu'))
model.add(Conv2D(32, 3, 3, border_mode='same', activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))

model.add(Conv2D(64, 3, 3, border_mode='same', activation='relu'))
model.add(Conv2D(64, 3, 3, border_mode='same', activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))

model.add(Conv2D(128, 3, 3, border_mode='same', activation='relu'))
model.add(Conv2D(128, 3, 3, border_mode='same', activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))

model.add(Conv2D(256, 3, 3, border_mode='same', activation='relu'))
model.add(Conv2D(256, 3, 3, border_mode='same', activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))

model.add(Flatten())
model.add(Dense(256, activation='relu'))
model.add(Dropout(0.5))

model.add(Dense(256, activation='relu'))
model.add(Dropout(0.5))

model.add(Dense(1))
model.add(Activation('sigmoid'))

model.compile(loss='binary_crossentropy',
            optimizer=RMSprop(lr=0.0001),
            metrics=['accuracy'])
```

```
with open(MODEL_SUMMARY_FILE,"w") as fh:
    model.summary(print_fn=lambda line: fh.write(line + "\n"))


# In[5]:


# Data scaling
training_data_generator = ImageDataGenerator(
    rescale=1./255,
    shear_range=0.1,
    zoom_range=0.1,
    horizontal_flip=True)
validation_data_generator = ImageDataGenerator(rescale=1./255)
test_data_generator = ImageDataGenerator(rescale=1./255)


# In[6]:


# Data preparation
training_generator = training_data_generator.flow_from_directory(
    training_data_dir,
    target_size=(IMAGE_WIDTH, IMAGE_HEIGHT),
    batch_size=BATCH_SIZE,
    class_mode="binary")
validation_generator = validation_data_generator.flow_from_directory(
    validation_data_dir,
    target_size=(IMAGE_WIDTH, IMAGE_HEIGHT),
    batch_size=BATCH_SIZE,
    class_mode="binary")
test_generator = test_data_generator.flow_from_directory(
    test_data_dir,
    target_size=(IMAGE_WIDTH, IMAGE_HEIGHT),
    batch_size=1,
    class_mode="binary",
    shuffle=False)


# In[7]:


# Training
model.fit_generator(
    training_generator,
    steps_per_epoch=len(training_generator.filenames) // BATCH_SIZE,
    epochs=EPOCHS,
    validation_data=validation_generator,
    validation_steps=len(validation_generator.filenames) // BATCH_SIZE,
    callbacks=[PlotLossesCallback(), CSVLogger(TRAINING_LOGS_FILE,
                                            append=False,
                                            separator=";")],
    verbose=1)
model.save_weights(MODEL_FILE)


# In[8]:


# Testing
probabilities = model.predict_generator(test_generator, TEST_SIZE)
for index, probability in enumerate(probabilities):
```

```
image_path = test_data_dir + "/" +test_generator.filenames[index]
img = mpimg.imread(image_path)
plt.imshow(img)
if probability > 0.5:
    plt.title("%.2f" % (probability[0]*100) + "% dog")
else:
    plt.title("%.2f" % ((1-probability[0])*100) + "% cat")
plt.show()
```

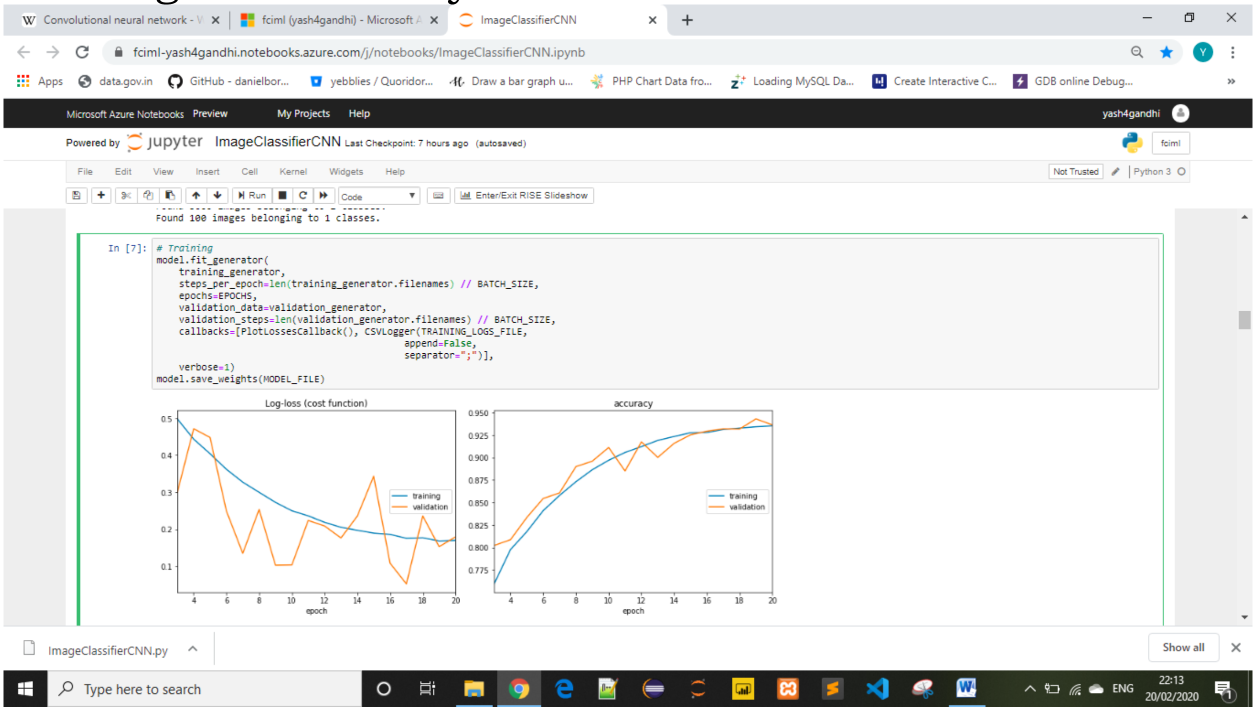
Output :

CNN Model :

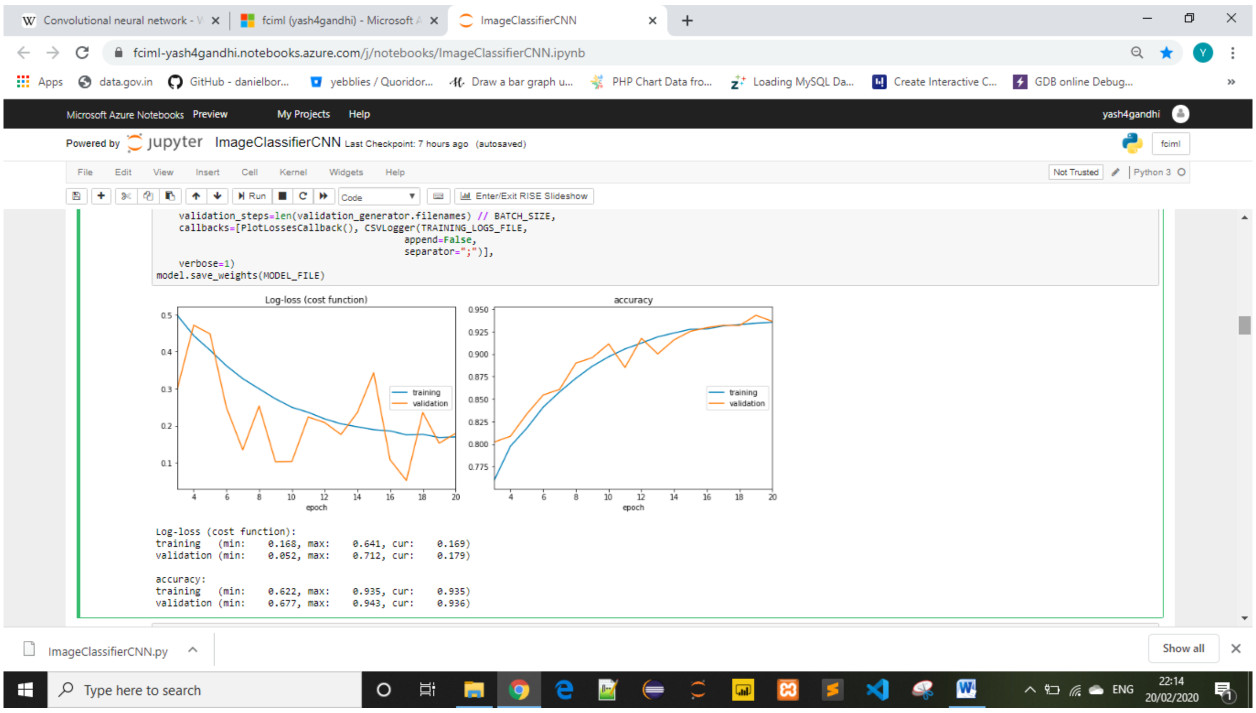


```
input_shape = (IMAGE_WIDTH, IMAGE_HEIGHT, 3)

In [4]: model = Sequential()

model.add(Conv2D(32, 3, 3, border_mode='same', input_shape=input_shape, activation='relu'))
model.add(Conv2D(32, 3, 3, border_mode='same', activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))

model.add(Conv2D(64, 3, 3, border_mode='same', activation='relu'))
model.add(Conv2D(64, 3, 3, border_mode='same', activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))

model.add(Conv2D(128, 3, 3, border_mode='same', activation='relu'))
model.add(Conv2D(128, 3, 3, border_mode='same', activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))

model.add(Conv2D(256, 3, 3, border_mode='same', activation='relu'))
model.add(Conv2D(256, 3, 3, border_mode='same', activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))

model.add(Flatten())
model.add(Dense(256, activation='relu'))
model.add(Dropout(0.5))

model.add(Dense(256, activation='relu'))
model.add(Dropout(0.5))

model.add(Dense(1))
model.add(Activation('sigmoid'))

model.compile(loss='binary_crossentropy',
            optimizer=RMSprop(lr=0.0001),
            metrics=['accuracy'])

with open(MODEL_SUMMARY_FILE,"w") as fh:
    model.summary(print_fn=lambda line: fh.write(line + "\n"))

/home/greg/.local/lib/python3.6/site-packages/ipykernel_launcher.py:4: UserWarning: Update your `Conv2D` call to the Keras 2 API: `Conv2D(32, (3, 3), input_shape=(200, 200,..., activation
```

Training and accuracy with loss curve



```
Found 100 images belonging to 1 classes.

In [7]: # Training
model.fit_generator(
    training_generator,
    steps_per_epoch=len(training_generator.filenames) // BATCH_SIZE,
    epochs=EPOCHS,
    validation_data=validation_generator,
    validation_steps=len(validation_generator.filenames) // BATCH_SIZE,
    callbacks=[PlotLossesCallback(), CSVLogger(TRAINING_LOGS_FILE,
                                            append=False,
                                            separator=";")],
    verbose=1)
model.save_weights(MODEL_FILE)
```

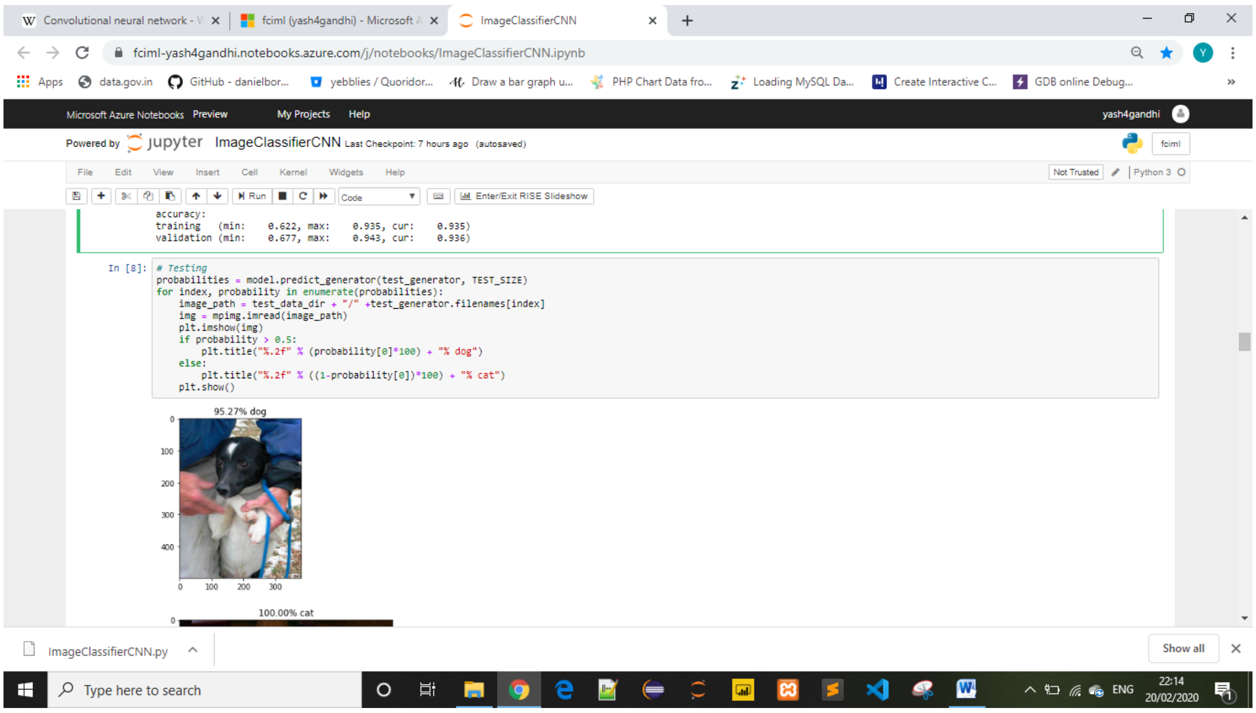


```
    validation_steps=len(validation_generator.filenames) // BATCH_SIZE,
    callbacks=[PlotLossesCallback(), CSVLogger(TRAINING_LOGS_FILE,
                                               append=False,
                                               separator=";")],

    verbose=1)
model.save_weights(MODEL_FILE)
```

```
Log-loss (cost function):
training   (min:    0.168, max:    0.641, cur:    0.169)
validation (min:    0.052, max:    0.712, cur:    0.179)

accuracy:
training   (min:    0.622, max:    0.935, cur:    0.935)
validation (min:    0.677, max:    0.943, cur:    0.936)
```

fciml-yash4gandhi.notebooks.azure.com/j/notebooks/ImageClassifierCNN.ipynb
Microsoft Azure Notebooks Preview My Projects Help
jupyter ImageClassifierCNN Last Checkpoint: 7 hours ago (autosaved)
Log-loss (cost function)
accuracy
training
validation
epoch

Prediction of test data

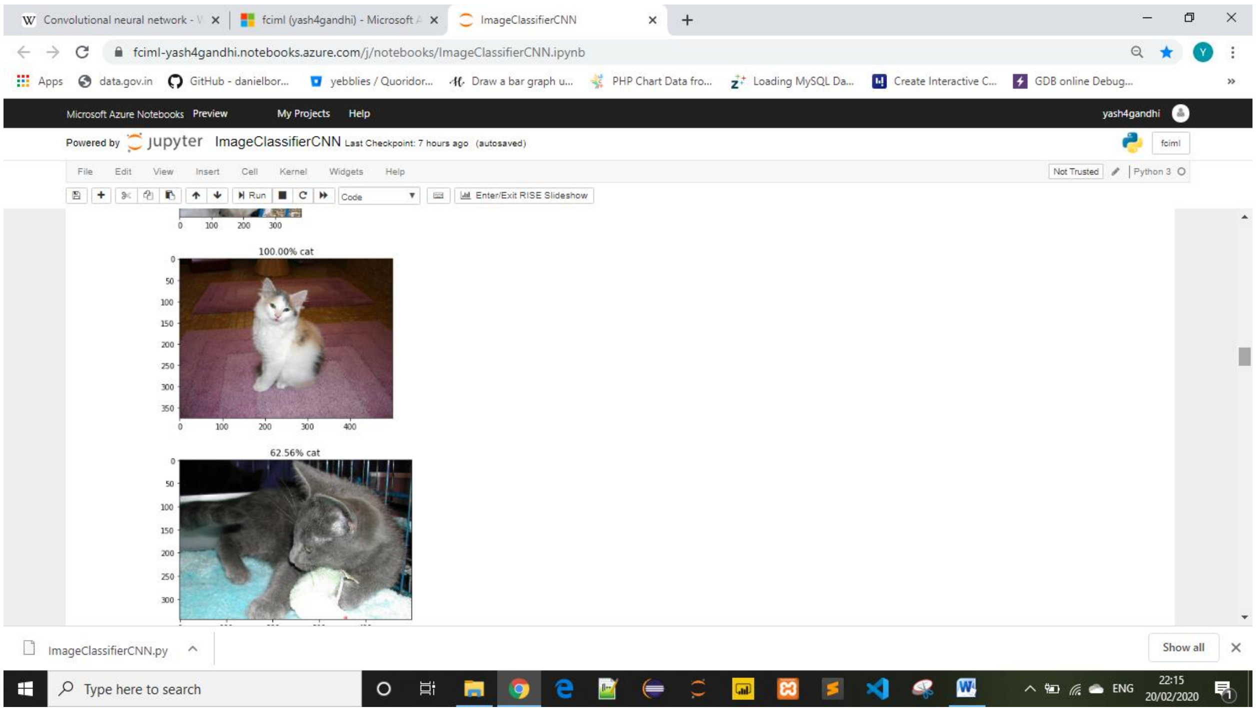


```
accuracy:
training   (min:    0.622, max:    0.935, cur:    0.935)
validation (min:    0.677, max:    0.943, cur:    0.936)
```

```
In [8]: # Testing
        probabilities = model.predict_generator(test_generator, TEST_SIZE)
        for index, probability in enumerate(probabilities):
            image_path = test_data_dir + "/" +test_generator.filenames[index]
            img = mpimg.imread(image_path)
            plt.imshow(img)
            if probability > 0.5:
                plt.title("%.2f" % (probability[0]*100) + "% dog")
            else:
                plt.title("%.2f" % ((1-probability[0])*100) + "% cat")
            plt.show()
```

fciml-yash4gandhi.notebooks.azure.com/j/notebooks/ImageClassifierCNN.ipynb
jupyter ImageClassifierCNN Last Checkpoint: 7 hours ago (autosaved)
95.27% dog
100.00% cat

Output


ImageClassifierCNN
fciml-yash4gandhi.notebooks.azure.com/j/notebooks/ImageClassifierCNN.ipynb
Microsoft Azure Notebooks Preview My Projects Help
jupyter ImageClassifierCNN Last Checkpoint: 7 hours ago (autosaved)
100.00% cat
62.56% cat
ImageClassifierCNN.py
Show all
Type here to search
22:15
20/02/2020

Conclusion :

We build a CNN image classification model for cat and dog data using keras library. Different hidden layers like Conv2D,MaxPooling,Dense,Dropout,Flatten were used with different number of parameters. Relu Activation function was used for all the hidden layers. Sigmoid Activation function was used for the output layer. With epoch size of 20 we achieved the accuracy of 95%.Test dataset contained variety of images.