SUMMER PROJECT REPORT

RECOGNIZING ARTIFICIAL THINGS AND NATURAL THINGS

SUBMITTED BY :

G RAVITEJA

ROLL NO:17R01A0420

B.TECH 2017 BATCH

Certificate

This is to certify that G RAVITEJA, enrolled in the B.TECH degree programme of the CMR Institute of Technology, Smart Bridge has successfully completed a summer project entitled 'Artificial vs Natural Recognizer using CNN' during the time period from 27th MAY, 2019 to 27th, 2019 under the guidance of Amarender Katkam chief technology officer, smart bridge.

_____

(Signature)

Amarender Katkam

Chief Technology Officer SmartBridge

ABSTRACT

Convolutional neural networks are primarily used to solve difficult image-driven pattern recognition tasks and with their precise yet simple architecture, offers a simplified method of getting started with ANNs. This document provides a brief introduction to CNNs.

Now-a-days, most of the AI scientists working on the autonomous cars which can drive itself without a driver and designing by using AI. The Autonomous cars can easily recognize the artificial and natural things in front of it such as cars, bikes, barrier gates, trees, animals, etc.., through computer vision.

INTRODUCTION

So basically what is CNN – as we know it's an machine learning algorithm for machines to understand the features of image with a foresight and remember the features to guess the name of the new image fed to the machine.

- Convolution layers consist of a set of learnable filters. Every filter has small width and height and the same depth as that of input volume (3 if the input layer is image input).
- In our project we chosen to run convolution on an image with dimension 64x64x3. Possible size of filters can be axax3, where 'a' can be 3, 5, 7, etc but small as compared to image dimension.
- During forward pass, we slide each filter across the whole input volume step by step where each step is called stride (which can have value 2 or 3 or even 4 for high dimensional images) and compute the dot product between the weights of filters and patch from input volume.
- As we slide our filters we'll get a 2-D output for each filter and we'll stack them together and as a result, we'll get output volume having a depth equal to the number of filters. The network will learn all the filters.

METHODOLOGY

Importing library files:

```
import numpy as np
import keras
from keras.models import Sequential
from keras.layers import Dense,Conv2D,MaxPooling2D,Flatten
```

```
C:\Users\Bhavana\Anaconda3\lib\site-packages\h5py\__init__.py:36: FutureWarning: Conversion of the second argument of issubdtyp
e from `float` to `np.floating` is deprecated. In future, it will be treated as `np.float64 == np.dtype(float).type`.
  from ._conv import register_converters as _register_converters
Using TensorFlow backend.
```

The library file NumPy is used as a container for generic data comprising of an N-dimensional array object, the library file keras is used to import the models and layers that are required to built the convolutional neural networks.

Pre-processing:

```
from keras.preprocessing.image import ImageDataGenerator
train_datagen=ImageDataGenerator(rescale=1./255,shear_range=0.2,horizontal_flip=True)
```

```
x_train=train_datagen.flow_from_directory(r'C:\Users\Bhavana\Desktop\New folder (2)\imagesets\imagesets',
                                          target_size=(64,64),batch_size=50,class_mode='binary')
```

```
Found 577 images belonging to 2 classes.
```

```
x_train.class_indices
```

```
{'artificial': 0, 'natural': 1}
```

```
cnn_model=Sequential()
```

In this pre-processing we are rescaling the image which it takes and also can be flipped horizontally. The input dimensions are set to the 64*64 which is a target size. Here index of the artificial and natural are shown.

Input and Convolution layers:

```
cnn_model.add(Conv2D(32,3,3,input_shape=(64,64,3),activation='relu'))
```

```
C:\Users\Bhavana\Anaconda3\lib\site-packages\ipykernel_launcher.py:1: UserWarning: Update your `Conv2D` call to the Keras 2 AP
I: `Conv2D(32, (3, 3), input_shape=(64, 64, 3..., activation="relu")`
  """Entry point for launching an IPython kernel.
```

This layer holds the raw input of image with width 64, height 64 and depth 3. A CNN is a neural network with some convolutional layers (and some other layers). A convolutional layer has a number of filters that does convolutional operation.

Max-Pooling Layer:

```
cnn_model.add(MaxPooling2D(pool_size=(2,2)))
```

This layer is periodically inserted in the covnets and its main function is to reduce the size of volume which makes the computation fast reduces memory and also prevents from overfitting. Two common types of pooling layers are **max pooling** and **average pooling**. We are using a max pool with 2 x 2 filters.

Flattening:

```
cnn_model.add(Flatten())
```

It gets the output of the pooling layer, flattens all its structure to create a single long feature vector to be used by the dense layer for the final classification.

Hidden Layer:

```
cnn_model.add(Dense(175,activation='relu'))
```

This layer is regular neural network layer which takes input from the previous layer and computes the class scores and outputs the 1-D array of size equal to the number of classes.

Output Layer:

```
cnn_model.add(Dense(1,activation='sigmoid'))
```

Compilation:

```
cnn_model.compile(optimizer='adam',loss='binary_crossentropy',metrics=['accuracy'])
```

Training:

```
cnn_model.fit_generator(x_train,samples_per_epoch=1000,epochs=7,nb_val_samples=200)
```

```
C:\Users\Bhavana\Anaconda3\lib\site-packages\ipykernel_launcher.py:1: UserWarning: The semantics of the Keras 2 argument `steps
_per_epoch` is not the same as the Keras 1 argument `samples_per_epoch`. `steps_per_epoch` is the number of batches to draw fro
m the generator at each epoch. Basically steps_per_epoch = samples_per_epoch/batch_size. Similarly `nb_val_samples`->`validatio
n_steps` and `val_samples`->`steps` arguments have changed. Update your method calls accordingly.
  """Entry point for launching an IPython kernel.
C:\Users\Bhavana\Anaconda3\lib\site-packages\ipykernel_launcher.py:1: UserWarning: Update your `fit_generator` call to the Kera
s 2 API: `fit_generator(<keras_pre..., epochs=7, steps_per_epoch=20, validation_steps=200)`
  """Entry point for launching an IPython kernel.
```

```
Epoch 1/7
20/20 [==============================] - 9s 431ms/step - loss: 1.4169 - acc: 0.5117
Epoch 2/7
20/20 [==============================] - 6s 311ms/step - loss: 0.6353 - acc: 0.6595
Epoch 3/7
20/20 [==============================] - 6s 324ms/step - loss: 0.5554 - acc: 0.7372
Epoch 4/7
20/20 [==============================] - 7s 371ms/step - loss: 0.4839 - acc: 0.7905
Epoch 5/7
20/20 [==============================] - 7s 353ms/step - loss: 0.4335 - acc: 0.8097
Epoch 6/7
20/20 [==============================] - 7s 367ms/step - loss: 0.3787 - acc: 0.8411
Epoch 7/7
20/20 [==============================] - 7s 356ms/step - loss: 0.3595 - acc: 0.8424

<keras.callbacks.History at 0x2ac9e278a20>
```

SAVING MODEL:

```
cnn_model.save('artificial-natural.h5')
```

LOADING THE MODEL:

```
from keras.models import load_model
import numpy as np
import cv2
model=load_model('artificial-natural.h5')
```

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

PREDICTING:

```
from skimage.transform import resize
def detect(frame):
    try:
        img=resize(frame,(64,64))
        img=np.expand_dims(img,axis=0)
        if(np.max(img)>1):
            img=img/255.0
        prediction=model.predict(img)
        print(prediction)
        prediction_class=model.predict_classes(img)
        print(prediction_class)
    except AttributeError:
        print('shape not found')
```

READING THE INPUT:

```
frame=cv2.imread(r"C:\Users\Bhavana\Desktop\New folder (2)\imagesets\test\man-made-things-1-728 (1).jpg")
```

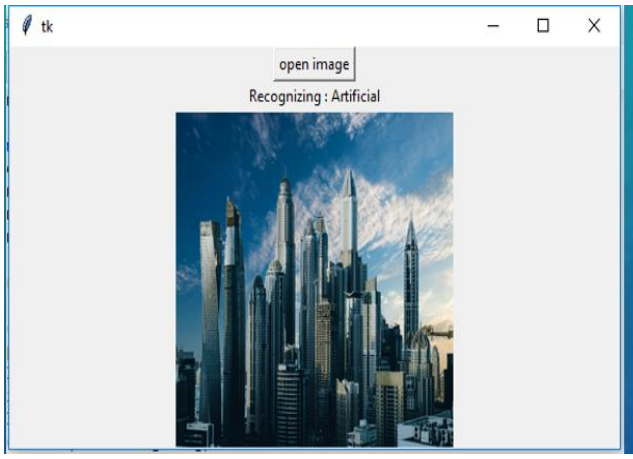
DETECTING THE INPUT:

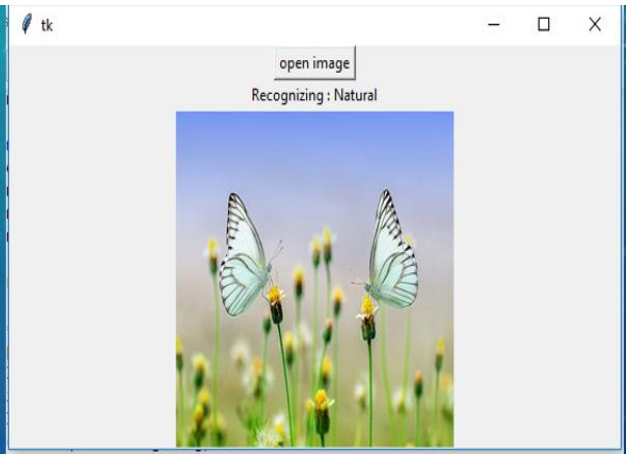
```
data=detect(frame)
```

```
[[0.5226122]]
[[1]]
```

```
C:\Users\Bhavana\Anaconda3\lib\site-packages\skimage\transform\_warps.py:84: UserWarning: The default mode, 'constant', will be changed to 'reflect' in skimage 0.15.
  warn("The default mode, 'constant', will be changed to 'reflect' in "
```

RESULT:


tk
open image
Recognizing : Artificial


tk
open image
Recognizing : Natural