Digital Image Processing Project

Cat and Dog image classification based on convolutional neural network using TensorFlow deep learning libraries in python

Prepared by: Muhammad Uzair Aslam

Project Report Submitted to:
Professor:Nicolo Adami

Department of Information Engineering
University of Brescia
Italy

Contents

Abstract This summary report is about convolutional neural network based image classifier using Tensorflow for image classification of cats and dogs. The strategy in this report is follow, first we shall give a short but brief introduction of convolutional neural network and TensorFlow. Then we shall discus the implementation and results. At the end we will include the references.

1 Introduction

Deep learning is a branch of artificial intelligence employing deep neural network architectures. Neural network are growing rapidly in every field of life rather they have almost dominates the whole artificial intelligence domain. Handling, learning, working using neural networks has becoming a fundamental need, mainly it's high performance and suitability to all the fields.NN are computer systems which are modeled based on the brains. They have capability to learn the given input and predict the output through an iterative process. All the processes of prediction are done by mathematical computation. How it works? In the given (Figure 1) we can easily see the a neural network process.

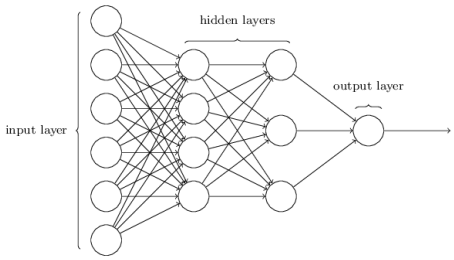

Figure 1: Neural Network

In the (figure:1) we can see three different layers, input, hidden and an output layer.Hidden layers can carry any number of neurons or nodes. The numbers of nodes in input layer has to be the same as the numbers of features in our problems.

Let see what exactly happing in each nodes. There is an example in (figure:2) let have a look on it.

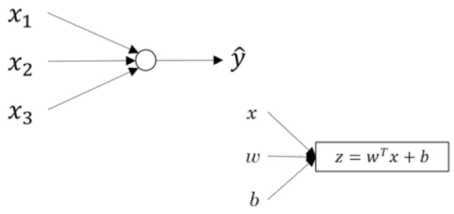

Figure 2: Neural Network Node

In the figure given node is input node, input features, (W) weight and (b) bias are taken as input and output (z) we can see in the image. These values are taken as matrices. Matrices makes computations easier. Secondly weight (W) and (b) bias are the random values generated in the fact of Gaussian distribution and also these values are use to compute the output of the nodes. With these values we are able to fit NN to our data.

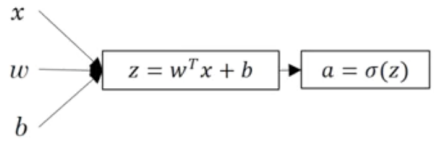

Figure 3: Neural Network Node Output

At the last getting the output (z) according to our (figure:3) shown above, the value (z) we use an activation function (Relu, Sigmoid) on z. These are the function we use to create our model in tensorflow, which we explain later in the coming sections. These function provide the non-linearity to to the model. if we don't use these model, then our output will be linear, then we will not be able to get the required results.
In final the hidden layers passes the data to other hidden layers, so the process continues and we get the output at the end according to the design of the model.

2 Convolutional Neural Network

Convolutional Neural Network like a combination of biology and math with a little bit computer science sprinkled. CNN is the most leading innovation in the field of computer vision. 2012 was the year when neural network got the fame as Alex krizhevsky used them to win that year's imageNet competition.

Facebook, google, Amazon, Pinsterset and instagram using Neural network to fulfill their needs in the race of data since. But the classic use of this network in image classification. So, our main focus will be on image classification, since we are going to use the CNN for image classification.

2.1 Image classification problems Human vs Computer

Classification of images is taking images which are belongs to the different classes like (Cats and Dogs etc) and giving them as input and getting the prediction/probability of the classes that best describe the images.
For human, recognition/classifying an image is different then a computer. The difference is shown in the (figure:4).

In the (figure:4) shown above is clearly describing the difference between the recognition of human and recognition of a computer. Computer see a image

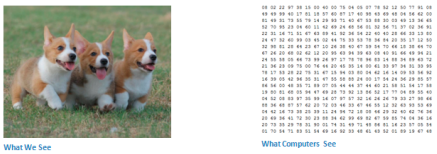

Figure 4: Human Vs Computer

as an array of pixel values, resolution and the size of the images. For example (32x32x3) array of numbers, the 3 refers to RED, GREEN and BLUE values. Pixel intensity values from 0 to 255 we describe, these number are important for computer.
On the other hand when we human see a picture, we generalize or classify the images from our past memories the way we learn about the objects. From example according to the figure:4. we can easily recognize that is a dog. Because we had learned since we start differentiate the objects from our childhood. we will look for the specific things like shape, legs, paws, etc.

2.2 Structure

As we explained in previous section how a computer and human recognize the images. In order to make a computer capable for this with CNN. We need a structure that we are going to explain. Main structure of the Convolutional Neural Network is based on the layers.These layers are CONVOLUTION, POOLING, and FULL CONNECTION layer. We can see these layers in the figure given below.

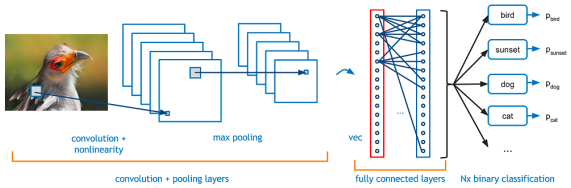

Figure 5: CNN Layers

convolutional Layer is use to extract features from an input image. Convolution keeps the relationship between pixels but learning image features using small squares of input data. Let's assume in the figure:6 given below a flash light that is shining over the top of left image. This flash light cover the area 5X5. Imagine this flash light move across all the area of input image. In machine learning term, this flash light called filter or something referred as a neuron or kernel, and the area of the image that shine is called receptive field.

Let's go little bit more deeper, if we look at the figure:7, there we have and image matrix (Volume) of dimension $(h \times w \times d)$ and a filter $f_h \times f_w \times d$. Output

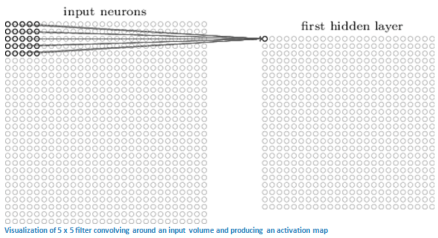

Figure 6: CNN Layers explanation

a volume dimension is $(h - f_h + 1) \times (w - f_h + 1) \times 1$.

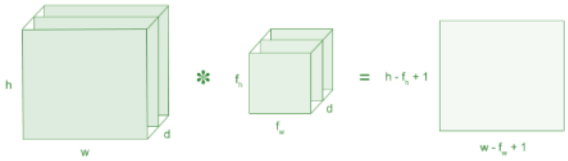

Figure 7: CNN Layers explanation

A 5×5 whose image pixel values are 0,1 and filter matrix 3×3 as shown in the figure:8.

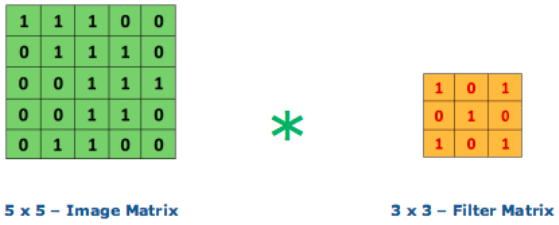

Figure 8: CNN Layers explanation

Pooling Layer section would reduce the number of parameters when the images are too large. Mostly used form of pooling is Max pooling that take the largest elements from the rectified feature map. In the figure:9 we can see the Max pooling.

Fully Connected Layer if each neuron in a layer receives input from all the neurons in previous layer, then this is called fully connected layer. Output of this layer is computed by matrix multiplication followed by bias offset.

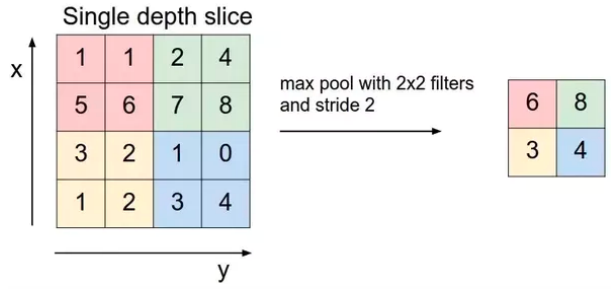

Figure 9: Max pooling

3 TensorFlow

TensorFlow is an open source software library for high performance numerical computation. It's flexible architecture allows easy deployment of computation across a variety of platforms (CPUs, GPUs, TPUs). Originally developed by researchers and engineers from the Google Brain team within Google's AI organization, it comes with strong support for machine learning and deep learning and the flexible numerical computation core is used across many other scientific domains.

3.1 Implementation

As we discuss before our main goal is image classification and we have the collection of Cats and Dogs images. Now we will implement our convolutional neural network with tensorflow.But we will use also the **tflearn**. TFLearn is a modular and transpernt deep learning library built on top of tensorflow. It was designed to provide a higher-level API to TensorFlow in order to facilitate and speed-up experimentations, while remaining fully transparent and compatible with it.

TFlearn is easy-to-use and understand high level API for implementing deep neural network, fast prototyping through highly modular built-in neural network layers, regularizers, optimizers, metrics. Full transparency over Tensorflow. All functions are built over tensors and can be used independently of TFLearn. Powerful helper functions to train any TensorFlow graph, with support of multiple inputs, outputs and optimizers. Easy and beautiful graph visualization, with details about weights, gradients, activations and more. Effortless device placement for using multiple CPU/GPU.

System details: we have use windows10 and anaconda in order to create our network, programing language we use is python.

3.2 Coding

First of all we import our database. we can see the code in figure:10.

```
#simply we loaded the images from our data base.
#-----------------------------------------------#

import cv2                # we use this library to resizing, images
import numpy as np        # used for dealing with arrays
import os                 # dealing with directories
from random import shuffle # mixing up or currently ordered data that might lead our network astray in training.
from tqdm import tqdm     # For Percentage bar's which are shown the code.

TRAIN_DIR = 'train'     #data
TEST_DIR = 'test'       #data
IMG_SIZE = 50
LR = 1e-3                 # Learning Rate = 0.001

uzair_model = 'dogsvscats-{}-{}.model'.format(LR, '2conv-basic') # For our memory which saved model is which.
```

Figure 10: Importing the images

Here we label our images $[1, 0]$ for the cats and $[0, 1]$ for the dogs. we can see the code in figure:11.

```
#Here we defined a funtion to label the Images.
#-----------------------------------------------#

def label_img(img):
    word_label = img.split('.')[-3]

    if word_label == 'cat': return [1,0]

    elif word_label == 'dog': return [0,1]
```

Figure 11: Image Labeling

In this part we create a function to train or database according to our labels. In the figure:12 we can see the both function.

```
#Here we defined a funtion to create train data.
#-----------------------------------------------#

def create_train_data():
    training_data = []
    for img in tqdm(os.listdir(TRAIN_DIR)):
        label = label_img(img)
        path = os.path.join(TRAIN_DIR,img)
        img = cv2.imread(path,cv2.IMREAD_GRAYSCALE)
        img = cv2.resize(img, (IMG_SIZE,IMG_SIZE))
        training_data.append([np.array(img),np.array(label)])
    shuffle(training_data)
    np.save('train_data.npy', training_data)
    return training_data
```

```
#Here we defined a funtion to create test data.
#-----------------------------------------------#


def process_test_data():
    testing_data = []
    for img in tqdm(os.listdir(TEST_DIR)):
        path = os.path.join(TEST_DIR,img)
        img_num = img.split('.')[0]
        img = cv2.imread(path,cv2.IMREAD_GRAYSCALE)
        img = cv2.resize(img, (IMG_SIZE,IMG_SIZE))
        testing_data.append([np.array(img), img_num])

    shuffle(testing_data)
    np.save('test_data.npy', testing_data)
    return testing_data
```

Figure 12: Training of Data

Output of our training function, we can see in the figure:13 also the percentage bars as well, which we got with the help of "tqdm" libraries.

```
train_data = create_train_data()
test_data = process_test_data()
```

```
100%|██████████| 25000/25000 [00:44<00:00, 558.68it/s]
100%|██████████| 12500/12500 [00:24<00:00, 506.07it/s]
```

Figure 13: Output of Training Data

In this section of the code we design our convolutional neural network. As we can see in the figure:14 that we define all the layer step by step the way we explain in the previous section of convolutional neural network introduction. But there are some Built-in operations that we have use for our network like Relu, softmax. There are more bt we just use these. ReLU stands for Rectified Linear Unit for a non-linear operation. softmax compute softmax activations.

Then our model where we added the arguments according to our need and we trained it.

```
import tflearn
from tflearn.layers.conv import conv_2d, max_pool_2d
from tflearn.layers.core import input_data, dropout, fully_connected
from tflearn.layers.estimator import regression
```

```
curses is not supported on this machine (please install/reinstall curses for an optimal experience)
```

```
import tensorflow as tf
tf.reset_default_graph()
convnet = input_data(shape=[None, IMG_SIZE, IMG_SIZE, 1], name='input')

convnet = conv_2d(convnet, 32, 5, activation='relu')
convnet = max_pool_2d(convnet, 5)

convnet = conv_2d(convnet, 64, 5, activation='relu')
convnet = max_pool_2d(convnet, 5)

convnet = conv_2d(convnet, 128, 5, activation='relu')
convnet = max_pool_2d(convnet, 5)

convnet = conv_2d(convnet, 64, 5, activation='relu')
convnet = max_pool_2d(convnet, 5)

convnet = conv_2d(convnet, 32, 5, activation='relu')
convnet = max_pool_2d(convnet, 5)

convnet = fully_connected(convnet, 1024, activation='relu')
convnet = dropout(convnet, 0.8)

convnet = fully_connected(convnet, 2, activation='softmax')
convnet = regression(convnet, optimizer='adam', learning_rate=LR, loss='categorical_crossentropy', name='targets')

model = tflearn.DNN(convnet, tensorboard_dir='log')



if os.path.exists('{}.meta'.format(uzair_model)):
    model.load(uzair_model)
    print('model loaded!')

train = train_data[:-500]
test = train_data[-500:]

X = np.array([i[0] for i in train]).reshape(-1,IMG_SIZE,IMG_SIZE,1)
Y = [i[1] for i in train]

test_x = np.array([i[0] for i in test]).reshape(-1,IMG_SIZE,IMG_SIZE,1)
test_y = [i[1] for i in test]

model.fit({'input': X}, {'targets': Y}, n_epoch=50, validation_set=({'input': test_x}, {'targets': test_y}),
    snapshot_step=500, show_metric=True, run_id=uzair_model)
model.save(uzair_model)
```

Figure 14: Model

After running our model we got the results around 98%. we can see in the figure:15

```
Training Step: 19149  | total loss: 0.05086 | time: 525.730s
| Adam | epoch: 050 | loss: 0.05086 - acc: 0.9819 -- iter: 24448/24500
Training Step: 19150  | total loss: 0.04942 | time: 528.085s
| Adam | epoch: 050 | loss: 0.04942 - acc: 0.9821 | val_loss: 1.64589 - val_acc: 0.7520 -- iter: 24500/24500
--
INFO:tensorflow:C:\Users\uzair.aslam\dogsvscats-0.001-2conv-basic-video.model is not in all_model_checkpoint_paths. Manually a
dding it.
```

Figure 15: Results

There is really cool feature in TensorFlow. We can create a "log" file of our model and we can see all the details on Tensorboard in the form of graphs. we

did the same is you see in the figure:14 we have created the log file in order to execute we use some commends shown in the figure:16.

```
#For the plots and Graphs

#tensorboard --logdir=foo:C:\Users\uzair.aslam\log
#http://NB-UASLAM:6006
```

Figure 16: For Graphs

4 Results

At the end here are the results the accuracy and the losses we have got after running.

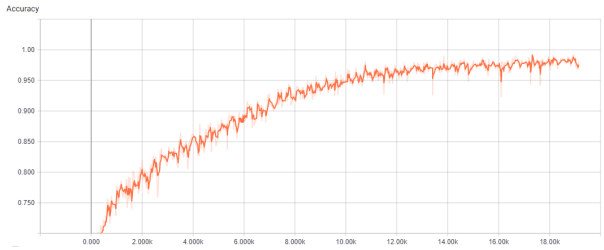

Figure 17: Accuracy

we have received almost 98% accuracy.

Our Raw accuracy

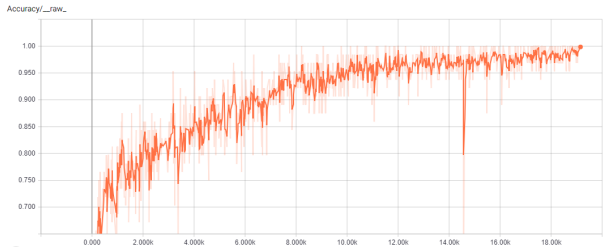

Figure 18: Raw Accuracy

Our validation accuracy

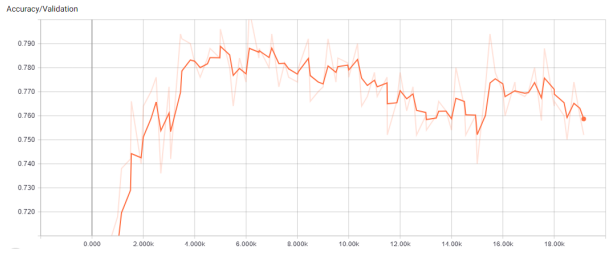

Figure 19: Validation Accuracy

Our losses which diereses gradually as we can see in the figure.

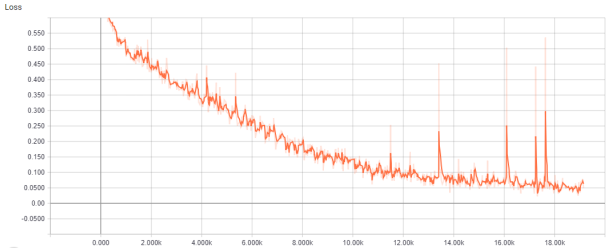

Figure 20: Losses

Our Raw losses

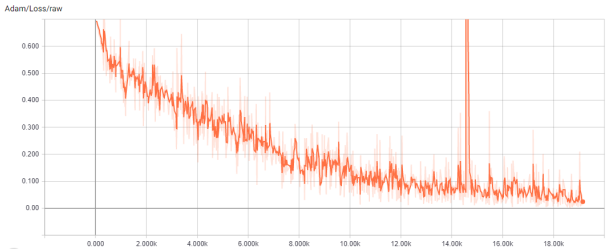

Figure 21: Raw Losses

Our Validation Losses

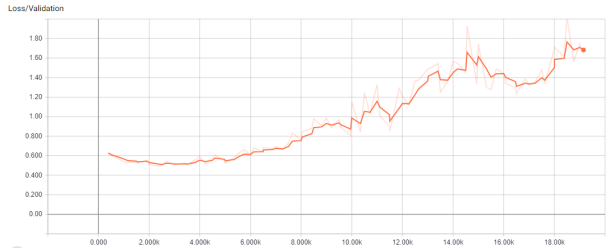

Figure 22: Validation Losses

At the end we have plot some images after training the data.

```
import matplotlib.pyplot as plt

test_data = np.load('test_data.npy')

fig=plt.figure()

for num,data in enumerate(test_data[:12]):
    # cat: [1,0]
    # dog: [0,1]

    img_num = data[1]
    img_data = data[0]

    y = fig.add_subplot(3,4,num+1)
    orig = img_data
    data = img_data.reshape(IMG_SIZE,IMG_SIZE,1)

    model_out = model.predict([data])[0]

    if np.argmax(model_out) == 1: str_label='Dog'
    else: str_label='Cat'

    y.imshow(orig,cmap='gray')
    plt.title(str_label)
    y.axes.get_xaxis().set_visible(False)
    y.axes.get_yaxis().set_visible(False)
plt.show()
```

Figure 23: Images after training

we have received some good results and also some losses as well. These kind of model we also can use for other types of images and field of life.