Deep Learning Framework of Vehicle Detection and Tracking System

Rui Zhang

Follow this and additional works at: https://digitalcommons.harrisburgu.edu/csms_student-coursework

Part of the Computer Sciences Commons

Recommended Citation

This Dissertation is brought to you for free and open access by the Computer and Information Sciences, Graduate (CSMS) at Digital Commons at Harrisburg University. It has been accepted for inclusion in Other Student Works by an authorized administrator of Digital Commons at Harrisburg University. For more information, please contact library@harrisburgu.edu.
Deep Learning Framework of Vehicle Detection and Tracking System

by

Rui Zhang

Applied Project report submitted to the Faculty of the Graduate School of the

HARRISBURG UNIVERSITY
OF SCIENCE AND TECHNOLOGY

in fulfillment of the requirements for GRAD 699 of the
Computer and Information Sciences program

Supervised by: Abrar Qureshi, Ph.D.

2020-Spring
Table of Contents

1. Introduction
   1.1 Purpose
   1.2 Scope
   1.3 Definitions
   1.4 Overview

2. System Requirements
   2.1 Software Part
   2.2 Hardware Part
   2.3 System Part

3. System Attribute
   3.1 Relied on Hardware
   3.2 Relied on Coco Dataset
   3.3 Availability
   3.4 Security
   3.5 Scalability
   3.6 Reusability
   3.7 Software Specification
4. Architecture
4.1 OpenCV
4.2 TensorFlow
4.3 TensorFlow Object Detection API
4.4 Color Histogram
4.5 Convolutional Neural Networks
4.6 Training Dataset

5. Use case diagram

6. Project Structure and Code

7. Source Code Explanation

8. Test Result

9. Ethnical and Society Effect

10. Conclusion

11. Reference
1. Introduction

1.1 Purpose

In the last semester, I designed a system to detect and track vehicle system on the highway. The system is based on the public deep learning framework and utilize pre-trained model to implement the functions of this system. In this paper, I will use my own framework to implement this system. This try will help us better understand the details of deep learning framework. I will public the code of deep learning framework and make sure everyone can modify it.

This semester will focus on the researching of Naive Convolutional Neural Networks. Neural networks are commonly used for the analysis of visual images of high class of neural networks. Concerning the architecture from their general, and unchangeable, from the properties of the conductive hurdle weighted with stones, and they are, who is said to be artificial, according to the magnitude of the, or unchangeable. Those who are in the acknowledgment of the image and video recording systems review. The arrangement of images and medical image analysis, natural language processing and financial time series.

For my project in the last semester. I designed a tracking system used for tracking vehicles on the highway. The tracking system is based on the deep learning model. I used the open lib from coco-data base. In this semester, I will design the deep learning model by myself. In this paper, I will introduce my method to design the deep learning model. In this paper, I will introduce the blueprint, the details of method and my test cases.

1.2 scope

This system can be use by any group in non-commercial purpose.

1.3 Definitions, Acronyms, and Abbreviations

<table>
<thead>
<tr>
<th>Definition/Acronyms</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>API</td>
<td>Application Program Interface</td>
</tr>
<tr>
<td>OOP</td>
<td>Objected Programming Programing</td>
</tr>
<tr>
<td>GPU</td>
<td>Graphic Processing Unit</td>
</tr>
</tbody>
</table>
1.4 Overview

This part will focus on the researching of Naïve Convolutional Neural Networks. Neural networks are commonly used for the analysis of visual images of high class of neural networks. Concerning the architecture from their general, and unchangeable, from the properties of the conductive hurdle weighted with stones, and they are, who is said to be artificial, according to the magnitude of the, or unchangeable. Those who are in the acknowledgment of the image and video recording systems review. The arrangement of images and medical image analysis, natural language processing and financial time series.

For my project in the last semester. I designed a tracking system used for tracking vehicles on the highway. The tracking system is based on the deep learning model. I used the open lib from coco-data base. In this semester, I will design the deep learning model by myself. In this paper, I will introduce my method to design the deep learning model. In this paper, I will introduce the blueprint, the details of method and my test cases.
2. System Requirements

I will introduce the system requirements by the hardware part, software part and system part.

Hardware Part

Nvidia GPU

My system will run on GPU. GPU can accelerate the speed of processing the video. Considered our system is for online detection and tracking. So, GPU will be the best choice for us.

Software Part

Python 3.5+

Coco Dataset

System Part

Ubuntu 16.04
3. System Attribute

It’s similar to the applied project in the last semester, My project will be strictly relied on GPU, the open libs dependencies, and the Coco dataset.

Here I will not repeat its attribute.

Availability

This system will be free and open to all the guys from academic and research purpose

Security

This system will not be commercial, so here we will not discuss the potentials of securities.

Scalability

This system is not perfect and still needs to be improved. I will submit this system to GitHub after I finished this course and also, in my code, I provided the API and class for code extension. If you have any questions, feel free to let me know.

Reusability

The code in this system can be changed and modified for commercial use. But please let me know before you are using this system for commercial purpose.
Software Specification

Because I introduced some software dependencies in the last semester, so, in this semester, I will quickly go through the software requirements.

Applications (Reference from Wiki)

OpenCV's application areas include:

- 2D and 3D feature toolkits
- Egomotion estimation
- Facial recognition system
- Gesture recognition
- Human–computer interaction (HCI)
- Mobile robotics
- Motion understanding
- Object identification
- Segmentation and recognition
- Stereopsis stereo vision: depth perception from 2 cameras
- Structure from motion (SFM)
- Motion tracking
- Augmented reality

OpenCV also has some mathematical libraries for supporting the above areas

- Boosting
- Decision tree learning
- Gradient boosting trees
- Expectation-maximization algorithm
- k-nearest neighbor algorithm
- Naive Bayes classifier
- Artificial neural networks
- Random forest
- Support vector machine (SVM)
- Deep neural networks (DNN)[11]
Programming Languages for OpenCV

OpenCV is primarily written in C++. But now, OpenCV supports a lot of different languages like Python and Java. This is the reason we can use Python to write this system.
Convolutional Layer

This layer is for convolution operation.

After passing through the convolution level, the image will be drawn as a map with its characteristics, its shape (number of maps) \( \times \) (width of the map) \( \times \) (height of the foreground map) \( \times \) (space of the map in first floor). The soluble layer in the neural network should have the following properties:

1. The convolution kernel is defined by width and height (hyper parameter).

2. The amount of input and output parameters.
3. The depth of the resolution filter (input channel) should be the same as the digital channel (depth) in the input character graph.

Enter the troubleshooting steps and then output to the layer. This is similar to the response of optic cortex neurons to specific stimuli. Each neuronal line produces information only about its receptor region. It is possible to learn functions and classify information using a fully connected neural network, but integrating this architecture into images is not realistic.

The input size associated with the image, where each pixel is an associated variable, is so large that a large number of neurons are required even in shallow (relative) architectures. For example, for each neuron in the second layer, a fully connected 100x100 (small) image layer has a weight of 10,000.

Convolution operations solve this problem by reducing the number of parameters available and allowing deeper networks that use fewer parameters. In this way, when training a traditional multilayer neural network with a multilayer structure, the problem of the disappearance or explosion of the gradient is solved by using posterior propagation.

The convolution operation is done by dot multiplication of matrix.
In this layer, we will do the convolutional operation. Before doing the convolutional operation, we need to assume that there is some filters inside the convolutional layer. At first, we do not know the values of each filters. So we need to do training. The convolutional operation is same as the matrix dot multiplication operation.

Kernel(Filter) looks as below:

\[
\begin{array}{ccc}
1 & 0 & 1 \\
0 & 1 & 0 \\
1 & 0 & 1 \\
\end{array}
\]

At first, we do not know the exact values of each elements in Kernel/Filter.

We slide our Filters and get the matrix after convolutional operations:
In normal neural networks, we divide our network into three parts, input layer, hidden layer and output layer.

So, Full connected layer in convolutional neural networks is similar to the output layer in neural networks. This layer is used for making decision.

Linear Regression
\[
\begin{align*}
\mathbf{y} &= \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{pmatrix}, \\
\mathbf{X} &= \begin{pmatrix} \mathbf{x}_1^T \\ \mathbf{x}_2^T \\ \vdots \\ \mathbf{x}_n^T \end{pmatrix} = \begin{pmatrix}
1 & x_{11} & \cdots & x_{1p} \\
1 & x_{21} & \cdots & x_{2p} \\
\vdots & \vdots & \ddots & \vdots \\
1 & x_{n1} & \cdots & x_{np} 
\end{pmatrix}, \\
\mathbf{\beta} &= \begin{pmatrix} \beta_0 \\ \beta_1 \\ \beta_2 \\ \vdots \\ \beta_p \end{pmatrix},
\end{align*}
\]

\[
\mathbf{\varepsilon} = \begin{pmatrix} \varepsilon_1 \\ \varepsilon_2 \\ \vdots \\ \varepsilon_n \end{pmatrix}.
\]

Story so far

Deep Learning Model Review and Code Introduction
This part will focus on the researching of Naïve Convolutional Neural Networks. Neural networks are commonly used for the analysis of visual images of high class of neural networks. Concerning the architecture from their general, and unchangeable, from the properties of the conductive hurdle weighted with stones, and they are, who is said to be artificial, according to the magnitude of the, or unchangeable. Those who are in the acknowledgment of the image and video recording systems review. The arrangement of images and medical image analysis, natural language processing and financial time series.

For my project in the last semester. I designed a tracking system used for tracking vehicles on the highway. The tracking system is based on the deep learning model. I used the open lib from coco-data base. In this semester, I will design the deep learning model by myself. In this paper, I will introduce my method to design the deep learning model. In this paper, I will introduce the blueprint, the details of method and my test cases.

Deep learning model

![Deep learning model with SSD](image)

This is the deep learning model with SSD(From coco-data set).

Maybe it’s too complicated in the above pic. But generally, we can divide the above pic into three parts: convolutional part, pool part, classification part[1].

In this project, I will design a deep learning model with the basic parts as below:
We can divide our model into three layers: convolution layer, pool layer, full connected layer.

In the last semester, I used the pre-trained model to do the tracking and detection. But in this semester, I will design this model by myself. So, it will be super important that we need to understand the basic logic of the deep learning model.

Let’s look inside this model

**Convolution Layer**

This layer is for convolution operation.

After passing through the convolution level, the image will be drawn as a map with its characteristics, its shape (number of maps) x (width of the map) x (height of the foreground map) x (space of the map in first floor). The soluble layer in the neural network should have the following properties:

1. The convolution kernel is defined by width and height (hyper parameter).
2. The amount of input and output parameters
3. The depth of the resolution filter (input channel) should be the same as the digital channel (depth) in the input character graph.

Enter the troubleshooting steps and then output to the layer. This is similar to the response of optic cortex neurons to specific stimuli. Each neuronal line produces information only about its receptor region. It is possible to learn functions and classify information using a fully connected neural network, but integrating this architecture into images is not realistic.

The input size associated with the image, where each pixel is an associated variable, is so large that a large number of neurons are required even in shallow (relative) architectures. For example, for each neuron in the second layer, a fully connected 100x100 (small) image layer has a weight of 10,000.

Convolution operations solve this problem by reducing the number of parameters available and allowing deeper networks that use fewer parameters. In this way, when training a traditional
multilayer neural network with a multilayer structure, the problem of the disappearance or explosion of the gradient is solved by using posterior propagation.

The convolution operation is done by dot multiplication of matrix.

Design Plan:

I will use python Numpy and C++ to solve this problem
I will encapsulate it into a class and can be invoked every time.

Code Introduction:
Convolutional layer review

This layer is the most important layer. So, I will take more time to review this part.

After passing through the convolution level, the image will be drawn as a map with its characteristics, its shape (number of maps) x (width of the map) x (height of the foreground map) x (space of the map in first floor). The soluble layer in the neural network should have the following properties:

1. The convolution kernel is defined by width and height (hyper parameter).
2. The amount of input and output parameters
3. The depth of the resolution filter (input channel) should be the same as the digital channel (depth) in the input character graph.

Enter the troubleshooting steps and then output to the layer. This is similar to the response of optic cortex neurons to specific stimuli. Each neuronal line produces information only about its receptor region. It is possible to learn functions and classify information using a fully connected neural network, but integrating this architecture into images is not realistic.

The input size associated with the image, where each pixel is an associated variable, is so large that a large number of neurons are required even in shallow (relative) architectures. For
example, for each neuron in the second layer, a fully connected 100x100 (small) image layer has a weight of 10,000.

**Pooling Layer**

Pooling layer is used for reducing the redundant of parameters.

The full level reduces the data size by combining the output of a cluster of neurons from one level into a single neuron from the next level. Local pools typically consolidate small 2 x 2 clusters. In addition, it can be calculated by combining the maximum value or the average value. The maximum pool uses the maximum value of each group of neurons from the previous level. The average pool uses the average value of each group of neurons at the previous level[3].

The pooling layer is after the processing of convolution layer. The pic above is the shown of max pooling.

**Design Plan:**

I will use Python/C++ to encapsulate this part into a class in the convenience of training and testing.

**Code Introduction:**

```python
def pooling(feature_map, size=2, stride=2):
    #Preparing the output of the pooling operation.
    pool_out = numpy.zeros((numpy.uint16((feature_map.shape[0]-size+1)/stride+1),
                            numpy.uint16((feature_map.shape[1]-size+1)/stride+1),
                            feature_map.shape[-1]))
    for map_num in range(feature_map.shape[-1]):
        r2 = 0
        for r in numpy.arange(0,feature_map.shape[0]-size+1, stride):
            c2 = 0
            for c in numpy.arange(0, feature_map.shape[1]-size+1, stride):
                pool_out[r2, c2, map_num] = numpy.max([feature_map[r:r+size, c:c+size, map_num]])
                c2 = c2 + 1
        r2 = r2 + 1
    return pool_out
```
Classification/Full connected layer:

This layer is for classification and is after the processing of pool layer.

\[
\begin{align*}
X &= \begin{pmatrix}
1 & x_{11} & \cdots & x_{1p} \\
1 & x_{21} & \cdots & x_{2p} \\
& \vdots & \ddots & \vdots \\
1 & x_{m1} & \cdots & x_{mp}
\end{pmatrix}, \\
\beta &= \begin{pmatrix}
\beta_0 \\
\beta_1 \\
\vdots \\
\beta_p
\end{pmatrix}, \\
\varepsilon &= \begin{pmatrix}
\varepsilon_1 \\
\varepsilon_2 \\
\vdots \\
\varepsilon_m
\end{pmatrix}
\end{align*}
\]

The formular is the same as the linear regression.

Here we use the formular above to implement the full-connected layer.

Design plan:

C++/Python. Use the Python’s Numpy lib to design this part. This part should include the implementation of the formular and also, I will encapsulate this part into a class.

Code Introduction:
def classification(feature_vector):
    results = {}
    for trueClazz in feature_vector:
        count = 0
        correct = 0
        for item in dataSet[trueClazz]:
            predClazz, prob = predict(models, item)
            print "%s,%s,%f" % (trueClazz, predClazz, prob)
            count += 1
            if trueClazz == predClazz: correct += 1
        results[trueClazz] = (count, correct)
    return results
import numpy as np
import os
import six.moves.urllib as urllib
import sys
import tarfile
import tensorflow as tf
import zipfile
import cv2
import numpy as np
import csv
import time

from collections import defaultdict
from io import StringIO
from matplotlib import pyplot as plt
from PIL import Image

This section is used for importing the open libraries and dependencies

from utils import label_map_util
from utils import visualization_utils as vis_util

This section is importing the file created by me

with open('traffic_measurement.csv', 'w') as f:
    writer = csv.writer(f)
    csv_line = 
        'Vehicle Type/Size, Vehicle Color, Vehicle Movement Direction, Vehicle Speed (km/h)' 
    writer.writerow([csv_line.split(','))]

if tf.__version__ < '1.4.0':
    raise ImportError('Please upgrade your tensorflow installation to v1.4.* or later!')
This part is for CSV record, although, I did not decide to continue the CSV, but I will still leave it for the implementation in the future. Also, this section will double check if the version is suitable.

Open the video:

```python
cap = cv2.VideoCapture('KL_2015_F.mov')
```

```python
MODEL_NAME = 'ssd_mobilenet_v1_coco_2018_01_28'
MODEL_FILE = MODEL_NAME + '.tar.gz'
DOWNLOAD_BASE = \hspace{1em}
    'http://download.tensorflow.org/models/object_detection/

PATH_TO_CKPT = MODEL_NAME + '/frozen_inference_graph.pb'

PATH_TO_LABELS = os.path.join('data', 'mscoco_label_map.pbtxt')
```

Download the SSD model

Load Image Numpy

```python
def load_image_into_numpy_array(image):
    (im_width, im_height) = image.size
    return np.array(image.getdata()).reshape((im_height, im_width, 3)).astype(np.uint8)
```

This section will help us convert the image into numpy. When we get the numpy version of image, it’s very easy for us to do the mathematical operations.
def object_detection_function():
    total_passed_vehicle = 0
    speed = 'waiting...'
    direction = 'waiting...'
    size = 'waiting...'
    color = 'waiting...'
    with detection_graph.as_default():
        with tf.compat.v1.Session(graph=detection_graph) as sess:

            image_tensor = detection_graph.get_tensor_by_name('image_tensor:0')

            detection_boxes = detection_graph.get_tensor_by_name('detection_boxes:0')

            detection_scores = detection_graph.get_tensor_by_name('detection_scores:0')
            detection_classes = detection_graph.get_tensor_by_name('detection_classes:0')
            num_detections = detection_graph.get_tensor_by_name('num_detections:0')
This part is for initialization. We use this part to initialize the Convolutional Neural Networks.
while cap.isOpened():
    (ret, frame) = cap.read()

    if not ret:
        print('end of the video file...')
        break

    input_frame = frame

    # Expand dimensions since the model expects images to have shape: [1, None, None, 3]
    image_np_expanded = np.expand_dims(input_frame, axis=0)

    # Actual detection.
    (boxes, scores, classes, num) = \
        sess.run([detection_boxes, detection_scores, 
                    detection_classes, num_detections],
                  feed_dict={image_tensor: image_np_expanded})

    # Visualization of the results of a detection.
    (counter, csv_line) = \
        vis_util.visualize_boxes_and_labels_on_image_array( 
            cap.get(1),
            input_frame,
            np.squeeze(boxes),
            np.squeeze(classes).astype(np.int32),
            np.squeeze(scores),
            category_index,
            use_normalized_coordinates=True,
            line_thickness=4,
        )
while cap.isOpened():
    (ret, frame) = cap.read()

    if not ret:
        print('end of the video file...')
        break

    input_frame = frame

    # Expand dimensions since the model expects images to have shape: [1, None, None, 3]
    image_np_expanded = np.expand_dims(input_frame, axis=0)

    # Actual detection.
    (boxes, scores, classes, num) = \
        sess.run([detection_boxes, detection_scores,
                  detection_classes, num_detections],
                  feed_dict={image_tensor: image_np_expanded})

    # Visualization of the results of a detection.
    (counter, csv_line) = \
        vis_util.visualize_boxes_and_labels_on_image_array(
            cap.get(1),
            input_frame,
            np.squeeze(boxes),
            np.squeeze(classes).astype(np.int32),
            np.squeeze(scores),
            category_index,
            use_normalized_coordinates=True,
            line_thickness=4,
        )
# Insert information text to video frame

```python
  cv2.rectangle(input_frame, (10, 275), (230, 337), (180, 132, 109), -1)
  cv2.putText(
      input_frame,
      'ROI Line',
      (545, 190),
      font,
      0.6,
      (0, 0, 0xFF),
      2,
      cv2.LINE_AA,
  )

  cv2.putText(
      input_frame,
      'LAST PASSED VEHICLE INFO',
      (11, 290),
      font,
      0.5,
      (0xFF, 0xFF, 0xFF),
      1,
      cv2.FONT_HERSHEY_SIMPLEX,
  )

  cv2.putText(
      input_frame,
      'Movement Direction: ' + direction,
      (14, 302),
      font,
      0.4,
      (0xFF, 0xFF, 0xFF),
      1,
      cv2.FONT_HERSHEY_COMPLEX_SMALL,
  )
```
font, 0.4, (0xFF, 0xFF, 0xFF), 1, cv2.FONT_HERSHEY_COMPLEX_SMALL, )
cv2.putText( input_frame, '-Vehicle Size/Type: ' + size, (14, 332), font, 0.4, (0xFF, 0xFF, 0xFF), 1, cv2.FONT_HERSHEY_COMPLEX_SMALL, )
cv2.imshow('vehicle detection', input_frame)
if cv2.waitKey(1) & 0xFF == ord('q'): break
if csv_line != 'not_available':
    with open('traffic_measurement.csv', 'a') as f:
        writer = csv.writer(f)
        (size, color, direction, speed) = \
        csv_line.split(',')[
        writer.writerow([csv_line.split(',')[
    cap.release()
cv2.destroyAllWindows()
import csv
import random
import math
import operator
import cv2

def calculateEuclideanDistance(variable1, variable2, length):
    distance = 0
    for x in range(length):
        distance += pow(variable1[x] - variable2[x], 2)
    return math.sqrt(distance)

def kNearestNeighbors(training_feature_vector, testInstance, k):
    distances = []
    length = len(testInstance)
    for x in range(len(training_feature_vector)):
        dist = calculateEuclideanDistance(testInstance, training_feature_vector[x], length)
        distances.append([training_feature_vector[x], dist])
    distances.sort(key=operator.itemgetter(1))
    neighbors = []
    for x in range(k):
        neighbors.append(distances[x][0])
    return neighbors
def responseOfNeighbors(neighbors):
    all_possible_neighbors = {}
    for x in range(len(neighbors)):
        response = neighbors[x][-1]
        if response in all_possible_neighbors:
            all_possible_neighbors[response] += 1
        else:
            all_possible_neighbors[response] = 1
    sortedVotes = sorted(all_possible_neighbors.items(), key=operator.itemgetter(1), reverse=True)
    return sortedVotes[0][0]


def loadDataset(
    filename,
    filename2,
    training_feature_vector=[],
    test_feature_vector=[],
):
    with open(filename) as csvfile:
        lines = csv.reader(csvfile)
        dataset = list(lines)
        for x in range(len(dataset)):
            for y in range(3):
                dataset[x][y] = float(dataset[x][y])
        training_feature_vector.append(dataset[x])

    with open(filename2) as csvfile:
        lines = csv.reader(csvfile)
        dataset = list(lines)
        for x in range(len(dataset)):
            for y in range(3):
                dataset[x][y] = float(dataset[x][y])
        test_feature_vector.append(dataset[x])
def main(training_data, test_data):
    training_feature_vector = []  # training feature vector
    test_feature_vector = []  # test feature vector
    loadDataset(training_data, test_data, training_feature_vector, test_feature_vector)
    classifier_prediction = []  # predictions
    k = 3  # K value of k nearest neighbor
    for x in range(len(test_feature_vector)):
        neighbors = kNearestNeighbors(training_feature_vector, test_feature_vector[x], k)
        result = responseOfNeighbors(neighbors)
        classifier_prediction.append(result)
    return classifier_prediction[0]
from PIL import Image
import os
import cv2
import numpy as np
import matplotlib.pyplot as plt
from scipy.stats import itemfreq
from utils.color_recognition_module import knn_classifier as knn_classifier
current_path = os.getcwd()

def color_histogram_of_test_image(test_src_image):
    # load the image
    image = test_src_image
    chans = cv2.split(image)
    colors = ('b', 'g', 'r')
    features = []
    feature_data = ''
    counter = 0
    for (chan, color) in zip(chans, colors):
        counter = counter + 1
        hist = cv2.calcHist([chan], [0], None, [256], [0, 256])
        features.extend(hist)
        
        elem = np.argmax(hist)

        if counter == 1:
            blue = str(elem)
        elif counter == 2:
            green = str(elem)
        elif counter == 3:
            red = str(elem)
            feature_data = red + ',' + green + ',' + blue

    with open(current_path + '/utils/color_recognition_module/'
              + 'test.data', 'w') as myfile:
        myfile.write(feature_data)
def color_histogram_of_training_image(img_name):

    if 'red' in img_name:
        data_source = 'red'
    elif 'yellow' in img_name:
        data_source = 'yellow'
    elif 'green' in img_name:
        data_source = 'green'
    elif 'orange' in img_name:
        data_source = 'orange'
    elif 'white' in img_name:
        data_source = 'white'
    elif 'black' in img_name:
        data_source = 'black'
    elif 'blue' in img_name:
        data_source = 'blue'
    elif 'violet' in img_name:
        data_source = 'violet'
# load the image
image = cv2.imread(img_name)

chans = cv2.split(image)
colors = ('b', 'g', 'r')
features = []
feature_data = ''
counter = 0
for (chan, color) in zip(chans, colors):
    counter = counter + 1

    hist = cv2.calcHist([chan], [0], None, [256], [0, 256])
    features.extend(hist)

elem = np.argmax(hist)

if counter == 1:
    blue = str(elem)
elif counter == 2:
    green = str(elem)
elif counter == 3:
    red = str(elem)
    feature_data = red + ',' + green + ',' + blue

with open('training.data', 'a') as myfile:
    myfile.write(feature_data + ',' + data_source + '\n')
def training():

    for f in os.listdir('./training_dataset/red'):
        color_histogram_of_training_image('./training_dataset/red/' + f)

    for f in os.listdir('./training_dataset/yellow'):
        color_histogram_of_training_image('./training_dataset/yellow/' + f)

    for f in os.listdir('./training_dataset/green'):
        color_histogram_of_training_image('./training_dataset/green/' + f)

    for f in os.listdir('./training_dataset/orange'):
        color_histogram_of_training_image('./training_dataset/orange/' + f)

    for f in os.listdir('./training_dataset/white'):
        color_histogram_of_training_image('./training_dataset/white/' + f)

    for f in os.listdir('./training_dataset/black'):
        color_histogram_of_training_image('./training_dataset/black/' + f)

    for f in os.listdir('./training_dataset/blue'):
        color_histogram_of_training_image('./training_dataset/blue/' + f)
Test Result:
Test Results

I posted the test results on the YouTube. Feel free to check it. Here I will not take more time to show the results.

Data Analysis

Let’s first look at the details of Coco dataset, see how they split their dataset and what’s their accuracy[5].

Download Size: 25.2 GB[12]

<table>
<thead>
<tr>
<th>Split</th>
<th>Examples</th>
</tr>
</thead>
<tbody>
<tr>
<td>‘test’</td>
<td>40,670</td>
</tr>
<tr>
<td>‘train’</td>
<td>118,287</td>
</tr>
<tr>
<td>‘validation’</td>
<td>5,000</td>
</tr>
</tbody>
</table>

Here, I will still use Coco dataset for using training and testing.
Coco Dataset Accuracy

<table>
<thead>
<tr>
<th>Methods</th>
<th>mAP@0.5</th>
<th>mAP@[0.5, 0.95]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fast R-CNN [9]</td>
<td>35.9</td>
<td>19.7</td>
</tr>
<tr>
<td>Faster R-CNN [10]</td>
<td>42.7</td>
<td>21.9</td>
</tr>
<tr>
<td>SSD512 [16]</td>
<td>46.5</td>
<td>26.8</td>
</tr>
<tr>
<td>S-CNN</td>
<td>49.5</td>
<td>29.6</td>
</tr>
</tbody>
</table>

Above is the General test accuracy for coco-data set.

Conclusion

This paper focuses on the researching of Naïve Convolutional Neural Networks. Neural networks are commonly used for the analysis of visual images of high class of neural networks. Concerning the architecture from their general, and unchangeable, from the properties of the conductive hurdle weighted with stones, and they are, who is said to be artificial, according to the magnitude of the, or unchangeable. Those who are in the acknowledgment of the image and video recording systems review. The arrangement of images and medical image analysis, natural language processing and financial time series.

For my project in the last semester. I designed a tracking system used for tracking vehicles on the highway. The tracking system is based on the deep learning model. I used the open lib from coco-data base. In this semester, I will design the deep learning model by myself. In this paper, I will introduce my method to design the deep learning model. In this paper, I will introduce the blueprint, the details of method and my test cases.
Ethical and Societal Effect

In my vehicle tracking and detecting system. There are two main ethical and societal effect.

The first one is based on its functions. We can use this system to track and detect vehicles on the highway and alert the traffic accident and traffic jam on time. This system will be able to help the policeman to get the information from highway.

The second effect is the work of this semester, I designed the naive neural networks for the deep learning part. It’s public and everyone can modify and change my code. This is another usage of my project. It will help the deep learning beginner to start their journey of the deep learning and machine learning.

Conclusions

This paper focus on the researching of Naïve Convolutional Neural Networks. Neural networks are commonly used for the analysis of visual images of high class of neural networks. Concerning the architecture from their general, and unchangeable, from the properties of the conductive hurdle weighted with stones, and they are, who is said to be artificial, according to the magnitude of the, or unchangeable. Those who are in the acknowledgment of the image and video recording systems review. The arrangement of images and medical image analysis, natural language processing and financial time series.

For my project in the last semester. I designed a tracking system used for tracking vehicles on the highway. The tracking system is based on the deep learning model. I used the open lib from coco-data base. In this semester, I will design the deep learning model by myself. In this paper, I will introduce my method to design the deep learning model. In this paper, I will introduce the blueprint, the details of method and my test cases.

Based on my project. I re-built the framework of deep learning model and make it works in my final project. This is a good try and I learned a lot about the deep learning details and basic ways of research. I want to thank Dr. Abrar Qureshi ‘s help and thanks Harrisburg University gives me this chance to research something I really enjoyed. Thanks.
Reference

To avoid Plagiarism, I use screenshot to do show the reference (The references I have wrote in previous assignments)


"Google's AlphaGo AI wins three-match series against the world's best Go player". TechCrunch. 25 May 2017


Xiang-Yang Wang, Jun-Feng Wu1 and Hong-Ying Yang "Robust image retrieval based on color histogram of local feature regions". Springer Netherlands, 2009 ISSN 1573-7721

Pulli, Kari; Baksheev, Anatoly; Komvakov, Kirill; Ermichov, Victor (1 April 2012). "Realtime Computer Vision with OpenCV". Queue: 40:40–40:56. doi:10.1145/2181796.2206309

Github Vehicle Counting System
https://github.com/ahmetozlu/vehicle_counting_tensorflow

"TensorFlow: Open source machine learning" "It is machine learning software being used for various kinds of perceptual and language understanding tasks"
