Mateusz Gościniak 22.01.2018

Image Recognition by TensorFlow with Python API.

Embedded Robotics
Witold Paluszyński, Ph.D.
Faculty of Electronics
Wrocław University of Science and Technology

Abstract

Image recognition with classification to 1 of 1000 categories.

The classification has been done by image-net.org.
The recognition problem was solved by 22 layers convocational deep neural network with fully-connected last layer. Model of CNN which was used in project - Inception v3, has been done by Google. Weights for CNN also has been counted by Google. All data of CNN model is public.
Dataset of image is accessed only for non-commercial use and mainly for student by image-net.org.

Keywords: Image recognition, Machine Learning, TensorFlow, Python 3.5

Software:

Windows 10 or Ubuntu 16.04

Python 3.5 (PSF License)
Python Libraries:
- TensorFlow 1.4 (Apache License 2.0)
- NumPy (BSD license)
- Pillow (MIT License)
- Matplotlib (PSF License)
- Jupyter notebook (BSD License)

PyCharm Community Edition (Apache License 2.0)

Introduction

Main goal:
- Preparing the TensorFlow environment to work
- Write program to recognition image data by use Inception v3 CNN model
- Perform tests
- Write program to recognition image data MNIST by own CNN model
- Perform tests

TensorFlow™ is an open source software library for numerical computation using data flow graphs.
https://www.tensorflow.org

So what's means TensorFlow name ? There is mix 2 idea: tensors and data flow. The tensor in shortcut is just generalization of matrix, vector and scalar. The data flow is paradigm in structural programming. This is really good framework to artificial intelligence and machine learning. Base concept is about build a graph (data flow) to compute and do Artificial Neural Network.

This task is about recognize what is on image by classify output to the 1 of 1000 categories. The training set, validation set is given by image-net.org:
http://image-net.org/challenges/LSVRC/2014/browse-synsets

About Artificial Neural Network (ANN)

Artificial neural networks (ANNs) or connectionist systems are computing systems inspired by the biological neural networks that constitute animal brains.
An ANN is based on a collection of connected units or nodes called artificial neurons (analogous to biological neurons in an animal brain). Each connection (analogous to a synapse) between artificial neurons can transmit a signal from one to another. The artificial neuron that receives the signal can process it and then signal artificial neurons connected to it.
[source: *https://en.wikipedia.org/wiki/Artificial_neural_network*]

Preparing the TensorFlow environment to work

On Windows 10 environment

Open link https://www.python.org/ftp/python/3.5.4/python-3.5.4-amd64-webinstall.exe
download and install it

Open the PowerShell as Administrator and write:
pip install tensorflow

pip install numpy
pip install pillow
pip install matplotlib
pip install jupyter

(optional) if you want use PyCharm Editor download and install it too.
https://www.jetbrains.com/pycharm-edu/download/download-thanks.html?platform=windows

On Ubuntu 16.04 LTE

In terminal write:
sudo pip install tensorflow
sudo pip install numpy
sudo pip install pillow
sudo pip install matplotlib
sudo pip install jupyter

(optional) if PyCharm Editor is needed, download and install it too.
https://www.jetbrains.com/pycharm-edu/download/download-thanks.html?platform=linux

Of course if you choose PyCharm you need to configure it properly.
On Windows and Linux is mostly the same:
Start create project: “File→New Project” and name it for example: “workspace”
Click on list name “Project Interpreter: ”
Choose (if isn’t by default) existing interpreter and write path to it.
In Windows it is “C:\Program Files\Python35\python.exe”
On Ubuntu it could be in “/usr/lib/python3.5”

Next create new *.py file in our workspace. “File→New”
Next choose python interpreter: “Run→Edit Configurations”
Click on green plus to add new configuration, name it and next write working directory and apply.
Write in main.py “print(‘Hello World’)” and run it to check if everything is ok.

Write program to recognition image data by use Inception v3 CNN model

Code snippet from jupyter notebook:

- Initialization of libraries

```
1  %matplotlib inline
2
3  import sys
4  import argparse
5  import matplotlib
6  import numpy as np
7  from PIL import Image
8  import matplotlib.pyplot as plt
9  import tensorflow as tf
```

- Providing paths

```
1  # Path to our image to check
2  path_to_image_1 = '.\\images\\jackfruit.jpg'
3  path_to_image_2 = '.\\images\\yogurt.jpg'
```

```
4
5   # There were provided with CNN model - Inception v3
6   path_to_labels = '.\\model\\imagenet_1000_labels.txt'
7   path_to_model = '.\\model\\inception_v3_2016_08_28_frozen.pb'
8   # frozen model means with const variables (already learned)
9
10  # There paths are in pb file, we can convert it to pbtxt and there
    are on first position and last
11  path_to_input_tensor = 'input'
12  path_to_output_tensor = 'InceptionV3/Predictions/Reshape_1'
13
14  # Adding import prefix is needed to load tensors
15  path_to_input_tensor = 'import/' + path_to_input_tensor
16  path_to_output_tensor = 'import/' + path_to_output_tensor
```

- Load graph

```
1   graph = tf.Graph() # Create graph to session
2   with open(path_to_model, "rb") as f: # Read binarry file
3       graph_def = tf.GraphDef()
4       graph_def.ParseFromString(f.read())
5   with graph.as_default(): # Use This graph as deafult in current
6   session
7       tf.import_graph_def(graph_def)
```

- Load image jpg and convert it to tensor

```
1   def convert_image_jpg_to_tensor(path_to_image):
2       file_reader = tf.read_file(path_to_image, 'file_reader')
3       image_reader = tf.image.decode_jpeg(file_reader, channels=3,
    name='jpeg_reader')
4       # image normalized
5       float_caster = tf.cast(image_reader, tf.float32)
6       dims_expander = tf.expand_dims(float_caster, 0)
7       resized = tf.image.resize_bilinear(dims_expander, [299, 299])
8       normalized = tf.divide(tf.subtract(resized, [0]), [255])
9       # run in session to resolve tensor, in TensorFlow, every
    operation have to be solved by running in session
10      sess = tf.Session()
11      tensor_from_image = sess.run(normalized)
12      return tensor_from_image
13
14  t1_image = convert_image_jpg_to_tensor(path_to_image_1)
15  t2_image = convert_image_jpg_to_tensor(path_to_image_2)
```

- Declare input and output operations on graph (CNN model)

```
1   input_operation = graph.get_operation_by_name(path_to_input_tensor);
2   output_operation =
    graph.get_operation_by_name(path_to_output_tensor);
```

- Running image on graph

```
with tf.Session(graph=graph) as sess:
    results = sess.run(output_operation.outputs[0],
{input_operation.outputs[0]: t1_image})
```

- Getting information with resoults

```
results = np.squeeze(results)
top_5_predictions = results.argsort()[-5:][::-1]
```

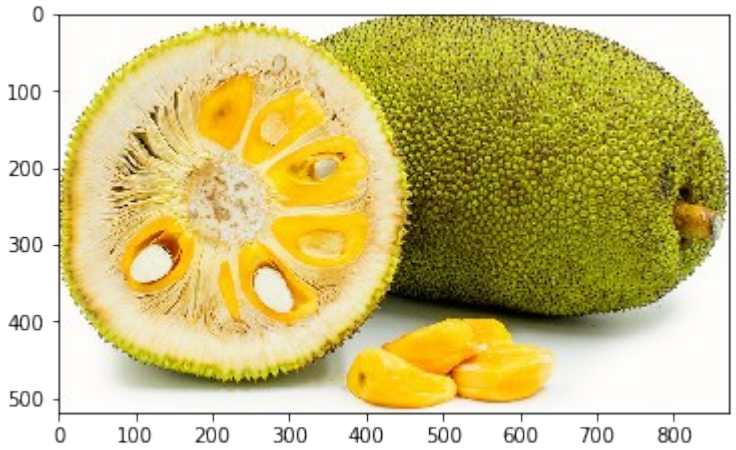
- Loading labels from file

```
labels = []
proto_as_ascii_lines = tf.gfile.GFile(path_to_labels).readlines()
for line in proto_as_ascii_lines:
    labels.append(line.rstrip())
```

- Finding exactly label

```
plt.imshow(Image.open(path_to_image_1))
plt.show()

for i in top_5_predictions:
    print(labels[i], ' - ', results[i]*100, '%')
```



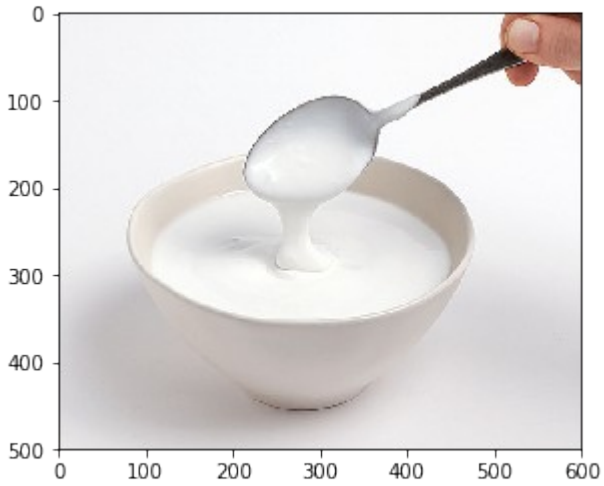

```
jackfruit  -  99.4270920753 %
drum  -  0.0650710717309 %
lemon  -  0.0330376526108 %
pineapple  -  0.0140951771755 %
orange  -  0.00623110900051 %
```

- Do it again to next prediction for image without label in dataset

```
with tf.Session(graph=graph) as sess:
    results = sess.run(output_operation.outputs[0],
{input_operation.outputs[0]: t2_image})

results = np.squeeze(results)
top_5_predictions = results.argsort()[-5:][::-1]
```

```
6
7   plt.imshow(Image.open(path_to_image_2))
8   plt.show()
9
10  for i in top_5_predictions:
11      print(labels[i], ' - ', results[i]*100, '%')
```




```
ladle  -  29.0698975325 %
mortar  -  16.8015018106 %
measuring cup  -  2.4708410725 %
soup bowl  -  2.26624701172 %
plate  -  2.12303288281 %
```

Write program to recognition image data MNIST by own CNN model

- Initialization of libraries

```
1   %matplotlib inline
2
3   import sys
4   import argparse
5   import matplotlib
6   import numpy as np
7   from PIL import Image
8   import matplotlib.pyplot as plt
9   import tensorflow as tf
10  from tensorflow.python.framework import graph_util
11  from tensorflow.python.platform import gfile
```

- Provide a one hot coding

```
1   def conv_to_one_hot(labels):
2       tab = np.zeros((labels.size, 10))
3       _i = 0
4       for label in labels:
5           if label == 0:
6               tab[_i] = [1, 0, 0, 0, 0, 0, 0, 0, 0, 0]
7           elif label == 1:
8               tab[_i] = [0, 1, 0, 0, 0, 0, 0, 0, 0, 0]
```

```
        elif label == 2:
            tab[_i] = [0, 0, 1, 0, 0, 0, 0, 0, 0, 0]
        elif label == 3:
            tab[_i] = [0, 0, 0, 1, 0, 0, 0, 0, 0, 0]
        elif label == 4:
            tab[_i] = [0, 0, 0, 0, 1, 0, 0, 0, 0, 0]
        elif label == 5:
            tab[_i] = [0, 0, 0, 0, 0, 1, 0, 0, 0, 0]
        elif label == 6:
            tab[_i] = [0, 0, 0, 0, 0, 0, 1, 0, 0, 0]
        elif label == 7:
            tab[_i] = [0, 0, 0, 0, 0, 0, 0, 1, 0, 0]
        elif label == 8:
            tab[_i] = [0, 0, 0, 0, 0, 0, 0, 0, 1, 0]
        elif label == 9:
            tab[_i] = [0, 0, 0, 0, 0, 0, 0, 0, 0, 1]
        _i += 1
    return tab
```

- Provided paths to pre created data set from http://yann.lecun.com/exdb/mnist
  Just converted from idx array to npy array

```
# load data
mnist_train_images = np.load('.\\MNIST\\train-images.npy')
mnist_test_images = np.load('.\\MNIST\\test-images.npy')
mnist_train_labels = conv_to_one_hot(np.load('.\\MNIST\\train-
labels.npy'))
mnist_test_labels = conv_to_one_hot(np.load('.\\MNIST\\test-
labels.npy'))
```

- Define help function to build graph

```
def conv2d(x, W):
    return tf.nn.conv2d(x, W, strides=[1, 1, 1, 1], padding='SAME')

def max_pool_2x2(x):
    return tf.nn.max_pool(x, ksize=[1, 2, 2, 1], strides=[1, 2, 2,
1], padding='SAME')


def weight_variable(shape):
    initial = tf.truncated_normal(shape, stddev=0.1)
    return tf.Variable(initial)


def bias_variable(shape):
    initial = tf.constant(0.1, shape=shape)
    return tf.Variable(initial)
```

- Build graph

```
# build network
# input
```

```
3   X = tf.placeholder(tf.float32, [None, 784], name='wejscie')
4   Y = tf.placeholder(tf.float32, [None, 10], name='labels')
5
6   # model
7   W_conv1 = weight_variable([5, 5, 1, 32])
8   b_conv1 = bias_variable([32])
9   x_image = tf.reshape(X, [-1, 28, 28, 1])
10  h_conv1 = tf.nn.relu(conv2d(x_image, W_conv1) + b_conv1)
11  h_pool1 = max_pool_2x2(h_conv1)
12
13  W_conv2 = weight_variable([5, 5, 32, 64])
14  b_conv2 = bias_variable([64])
15  h_conv2 = tf.nn.relu(conv2d(h_pool1, W_conv2) + b_conv2)
16  h_pool2 = max_pool_2x2(h_conv2)
17
18  W_fc1 = weight_variable([7 * 7 * 64, 1024])
19  b_fc1 = bias_variable([1024])
20  h_pool2_flat = tf.reshape(h_pool2, [-1, 7*7*64])
21  h_fc1 = tf.nn.relu(tf.matmul(h_pool2_flat, W_fc1) + b_fc1)
22
23  # keep_prob = tf.placeholder(tf.float32)
24  # h_fc1_drop = tf.nn.dropout(h_fc1, keep_prob)
25
26  W_fc2 = weight_variable([1024, 10])
27  b_fc2 = bias_variable([10])
28
29  # y_conv = tf.matmul(h_fc1_drop, W_fc2) + b_fc2
30  y_conv = tf.matmul(h_fc1, W_fc2) + b_fc2
31
32  final_tensor = tf.nn.softmax(y_conv, name='wyjscie')
33  cross_entropy =
    tf.reduce_mean(tf.nn.softmax_cross_entropy_with_logits(labels=Y,
    logits=y_conv))
34  train_step = tf.train.AdamOptimizer(1e-4).minimize(cross_entropy)
    correct_prediction = tf.cast(tf.equal(tf.argmax(y_conv, 1),
35  tf.argmax(Y, 1)), tf.float32)
36  accuracy = tf.reduce_mean(correct_prediction)
```

- Create function to batch return

```
1   def take_next_batch(data_set_x, data_set_y, _x):
2       size = data_set_x.__len__()
3       batch_x = np.zeros((_x, int(data_set_x.size / size)), int)
4       batch_y = np.zeros((_x, int(data_set_y.size / size)), int)
5       for _i in range(0, _x):
6           random = np.random.random_integers(0, size-1, 1)
7           batch_x[_i] = data_set_x[random]
8           batch_y[_i] = data_set_y[random]
9       return batch_x, batch_y
```

- Start Learning and save the graph

```
1   with tf.Session() as sess:
2       sess.run(tf.global_variables_initializer())
```

```
    for i in range(200): # should be 20000 but for demonstrate only the process is 200
        batch_xs, batch_ys = take_next_batch(mnist_train_images, mnist_train_labels, 50)
        if i % 100 == 0:
            # train_accuracy = accuracy.eval(feed_dict={X: batch_xs, Y: batch_ys, keep_prob: 1.0})
            train_accuracy = accuracy.eval(feed_dict={X: batch_xs, Y: batch_ys})
            print('step %d, training accuracy %g' % (i, train_accuracy))
            # print(sess.run(W_conv1[0])) # zobaczyć jakoś ?
        # train_step.run(feed_dict={X: batch_xs, Y: batch_ys, keep_prob: 0.5})
        train_step.run(feed_dict={X: batch_xs, Y: batch_ys})
    # print('test accuracy %g' % accuracy.eval(feed_dict={X: mnist_test_images, Y: mnist_test_labels, keep_prob: 1.0}))
    print('test accuracy %g' % accuracy.eval(feed_dict={X: mnist_test_images, Y: mnist_test_labels}))

    # save to frozen pb
    output_graph_def = graph_util.convert_variables_to_constants(sess, sess.graph.as_graph_def(), ['wyjscie'])
    with gfile.FastGFile('.\\model\\MNIST_frozen.pb', 'wb') as f:
        f.write(output_graph_def.SerializeToString())
```

step 0, training accuracy 0.12
step 100, training accuracy 0.82
test accuracy 0.8926
INFO:tensorflow:Froze 8 variables.
Converted 8 variables to const ops.

- After the learn, evaluating graph

```
def load_graph(model_file):
    graph = tf.Graph()
    with open(model_file, "rb") as f:
        graph_def = tf.GraphDef()
        graph_def.ParseFromString(f.read())
    with graph.as_default():
        tf.import_graph_def(graph_def)
    return graph


def load_image(file_name, raw=False, negative=False):
    result = np.zeros(784, np.ubyte)
    _i = 0
    with open(file_name, 'r') as file:
        if not raw:
            file.readline()
            file.readline()
            file.readline()
            file.readline()
```

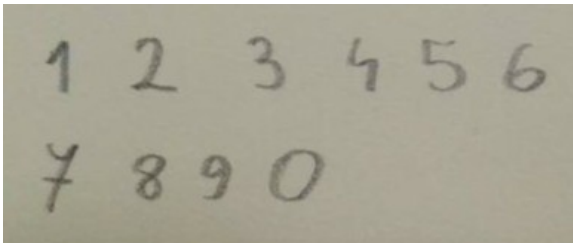
```
        lines = file.read().split('\n')
        for line in lines:
            if str.isnumeric(line):
                if negative:
                    result[_i] = int(line) * -1
                else:
                    result[_i] = int(line)
                _i += 1
    return result


def flow(_image):
    input_operation = graph.get_operation_by_name('import/wejscie')
    output_operation = graph.get_operation_by_name('import/wyjscie')
    _t = np.reshape(_image, (1, 28 * 28))
    with tf.Session(graph=graph) as _sess:
        _results = np.squeeze(_sess.run(output_operation.outputs[0], {input_operation.outputs[0]: _t}))
    return np.around(_results)

graph = load_graph('.\\model\\pretrained_MNIST_frozen.pb')

image = load_image('.\\images\\PGM\\0.pgm')
print('0 = ', flow(image))
image = load_image('.\\images\\PGM\\1.pgm')
print('1 = ', flow(image))
image = load_image('.\\images\\PGM\\2.pgm')
print('2 = ', flow(image))
image = load_image('.\\images\\PGM\\3.pgm')
print('3 = ', flow(image))
image = load_image('.\\images\\PGM\\4.pgm')
print('4 = ', flow(image))
image = load_image('.\\images\\PGM\\5.pgm')
print('5 = ', flow(image))
image = load_image('.\\images\\PGM\\6.pgm')
print('6 = ', flow(image))
image = load_image('.\\images\\PGM\\7.pgm')
print('7 = ', flow(image))
image = load_image('.\\images\\PGM\\8.pgm')
print('8 = ', flow(image))
image = load_image('.\\images\\PGM\\9.pgm')
print('9 = ', flow(image))
```

```
0 =  [ 1.  0.  0.  0.  0.  0.  0.  0.  0.  0.]
1 =  [ 0.  1.  0.  0.  0.  0.  0.  0.  0.  0.]
2 =  [ 0.  0.  1.  0.  0.  0.  0.  0.  0.  0.]
3 =  [ 0.  0.  0.  1.  0.  0.  0.  0.  0.  0.]
4 =  [ 0.  0.  0.  0.  1.  0.  0.  0.  0.  0.]
5 =  [ 0.  0.  0.  0.  0.  1.  0.  0.  0.  0.]
6 =  [ 0.  0.  0.  0.  0.  0.  1.  0.  0.  0.]
7 =  [ 0.  0.  0.  0.  0.  0.  0.  0.  1.  0.]
8 =  [ 0.  0.  0.  0.  0.  0.  0.  0.  1.  0.]
9 =  [ 0.  0.  0.  0.  0.  0.  0.  0.  0.  1.]
```



Summary

TensorFlow is easy in use and really good tool for learning.
Well, in case image classify to syn-sets/categories by CNN have the best results, but described technique is not so good to use in real world task because images has to be classify to categories by human. It is image recognition without learned.
In second test we could see how much important is data-set. In MNIST recognition problem we have only American version of 7, so European handwritten 7 is impossible to correctly detect

References

www.tensorflow.org
www.image-net.org
Going Deeper with Convolutions, *Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, Andrew Rabinovich*, https://arxiv.org/abs/1409.4842