

# Chapter\_12\_Section\_2\_Loading\_VGG\_Model

February 9, 2019

## 1 Using a previously trained tf model

### 1.1 Explanation

In this notebook a model is defined as a class and then the weights of the model parameters are loaded using the `load_weights` method

### 1.2 Image embedding

The VGG-16 TensorFlow port is by Davi Frossard (<http://www.cs.toronto.edu/~frossard/post/vgg16/>).  
Along with TensorFlow, it requires the following libraries:

```
$ pip install scipy
$ pip install Pillow
```

You will need to download the model parameters

```
$ wget https://www.cs.toronto.edu/~frossard/vgg16/vgg16_weights.npz
```

```
In [1]: #####
        # Davi Frossard, 2016
        # VGG16 implementation in TensorFlow
        # Details:
        # http://www.cs.toronto.edu/~frossard/post/vgg16/
        #
        # Model from https://gist.github.com/ksimonyan/211839e770f7b538e2d8#file-readme-md #
        # Weights from Caffe converted using https://github.com/ethereon/caffe-tensorflow #
        #####

        %matplotlib inline
        from matplotlib import pyplot as plt

        import tensorflow as tf
        import numpy as np
        from scipy.misc import imread, imresize
        from imagenet_classes import class_names
```

```

class vgg16:
    def __init__(self, imgs, weights=None, sess=None):
        self.imgs = imgs
        tf.summary.image("imgs", self.imgs)
        self.convlayers()
        self.fc_layers()
        tf.summary.histogram("fc2", self.fc2)
        self.probs = tf.nn.softmax(self.fc3l)
        if weights is not None and sess is not None:
            self.load_weights(weights, sess)

    def convlayers(self):
        self.parameters = []

        # zero-mean input
        with tf.name_scope('preprocess') as scope:
            mean = tf.constant([123.68, 116.779, 103.939],
                               dtype=tf.float32, shape=[1, 1, 1, 3],
                               name='img_mean')
            images = self.imgs - mean

        # conv1_1
        with tf.name_scope('conv1_1') as scope:
            kernel = tf.Variable(tf.truncated_normal([3, 3, 3, 64],
                                                    dtype=tf.float32,
                                                    stddev=1e-1),
                                name='weights')
            conv = tf.nn.conv2d(images, kernel, [1, 1, 1, 1], padding='SAME')
            biases = tf.Variable(tf.constant(0.0, shape=[64], dtype=tf.float32),
                                trainable=True, name='biases')
            out = tf.nn.bias_add(conv, biases)
            self.conv1_1 = tf.nn.relu(out, name=scope)
            self.parameters += [kernel, biases]

        # conv1_2
        with tf.name_scope('conv1_2') as scope:
            kernel = tf.Variable(tf.truncated_normal([3, 3, 64, 64],
                                                    dtype=tf.float32,
                                                    stddev=1e-1),
                                name='weights')
            conv = tf.nn.conv2d(self.conv1_1, kernel, [1, 1, 1, 1],
                                padding='SAME')
            biases = tf.Variable(tf.constant(0.0, shape=[64],
                                                    dtype=tf.float32),
                                trainable=True, name='biases')
            out = tf.nn.bias_add(conv, biases)
            self.conv1_2 = tf.nn.relu(out, name=scope)

```

```

        self.parameters += [kernel, biases]

# pool1
self.pool1 = tf.nn.max_pool(self.conv1_2,
                            ksize=[1, 2, 2, 1],
                            strides=[1, 2, 2, 1],
                            padding='SAME',
                            name='pool1')

# conv2_1
with tf.name_scope('conv2_1') as scope:
    kernel = tf.Variable(tf.truncated_normal([3, 3, 64, 128],
                                              dtype=tf.float32,
                                              stddev=1e-1),
                        name='weights')
    conv = tf.nn.conv2d(self.pool1, kernel, [1, 1, 1, 1],
                        padding='SAME')
    biases = tf.Variable(tf.constant(0.0, shape=[128],
                                         dtype=tf.float32),
                        trainable=True, name='biases')
    out = tf.nn.bias_add(conv, biases)
    self.conv2_1 = tf.nn.relu(out, name=scope)
    self.parameters += [kernel, biases]

# conv2_2
with tf.name_scope('conv2_2') as scope:
    kernel = tf.Variable(tf.truncated_normal([3, 3, 128, 128],
                                              dtype=tf.float32,
                                              stddev=1e-1),
                        name='weights')
    conv = tf.nn.conv2d(self.conv2_1, kernel,
                        [1, 1, 1, 1], padding='SAME')
    biases = tf.Variable(tf.constant(0.0, shape=[128],
                                         dtype=tf.float32),
                        trainable=True, name='biases')
    out = tf.nn.bias_add(conv, biases)
    self.conv2_2 = tf.nn.relu(out, name=scope)
    self.parameters += [kernel, biases]

# pool2
self.pool2 = tf.nn.max_pool(self.conv2_2,
                            ksize=[1, 2, 2, 1],
                            strides=[1, 2, 2, 1],
                            padding='SAME',
                            name='pool2')

# conv3_1
with tf.name_scope('conv3_1') as scope:

```

```

kernel = tf.Variable(tf.truncated_normal([3, 3, 128, 256],
                                         dtype=tf.float32,
                                         stddev=1e-1),
                    name='weights')
conv = tf.nn.conv2d(self.pool2, kernel,
                   [1, 1, 1, 1], padding='SAME')
biases = tf.Variable(tf.constant(0.0, shape=[256],
                                     dtype=tf.float32),
                    trainable=True, name='biases')
out = tf.nn.bias_add(conv, biases)
self.conv3_1 = tf.nn.relu(out, name=scope)
self.parameters += [kernel, biases]

# conv3_2
with tf.name_scope('conv3_2') as scope:
    kernel = tf.Variable(tf.truncated_normal([3, 3, 256, 256],
                                             dtype=tf.float32,
                                             stddev=1e-1),
                        name='weights')
    conv = tf.nn.conv2d(self.conv3_1, kernel,
                       [1, 1, 1, 1], padding='SAME')
    biases = tf.Variable(tf.constant(0.0,
                                     shape=[256], dtype=tf.float32),
                        trainable=True, name='biases')
    out = tf.nn.bias_add(conv, biases)
    self.conv3_2 = tf.nn.relu(out, name=scope)
    self.parameters += [kernel, biases]

# conv3_3
with tf.name_scope('conv3_3') as scope:
    kernel = tf.Variable(tf.truncated_normal([3, 3, 256, 256],
                                             dtype=tf.float32,
                                             stddev=1e-1),
                        name='weights')
    conv = tf.nn.conv2d(self.conv3_2, kernel,
                       [1, 1, 1, 1], padding='SAME')
    biases = tf.Variable(tf.constant(0.0,
                                     shape=[256],
                                     dtype=tf.float32),
                        trainable=True, name='biases')
    out = tf.nn.bias_add(conv, biases)
    self.conv3_3 = tf.nn.relu(out, name=scope)
    self.parameters += [kernel, biases]

# pool3
self.pool3 = tf.nn.max_pool(self.conv3_3,
                           ksize=[1, 2, 2, 1],
                           strides=[1, 2, 2, 1],

```

```

padding='SAME',
name='pool3')

# conv4_1
with tf.name_scope('conv4_1') as scope:
    kernel = tf.Variable(tf.truncated_normal([3, 3, 256, 512],
                                             dtype=tf.float32,
                                             stddev=1e-1),
                        name='weights')
    conv = tf.nn.conv2d(self.pool3, kernel,
                        [1, 1, 1, 1],
                        padding='SAME')
    biases = tf.Variable(tf.constant(0.0,
                                     shape=[512], dtype=tf.float32),
                        trainable=True, name='biases')
    out = tf.nn.bias_add(conv, biases)
    self.conv4_1 = tf.nn.relu(out, name=scope)
    self.parameters += [kernel, biases]

# conv4_2
with tf.name_scope('conv4_2') as scope:
    kernel = tf.Variable(tf.truncated_normal([3, 3, 512, 512],
                                             dtype=tf.float32,
                                             stddev=1e-1),
                        name='weights')
    conv = tf.nn.conv2d(self.conv4_1,
                        kernel, [1, 1, 1, 1],
                        padding='SAME')
    biases = tf.Variable(tf.constant(0.0,
                                     shape=[512], dtype=tf.float32),
                        trainable=True, name='biases')
    out = tf.nn.bias_add(conv, biases)
    self.conv4_2 = tf.nn.relu(out, name=scope)
    self.parameters += [kernel, biases]

# conv4_3
with tf.name_scope('conv4_3') as scope:
    kernel = tf.Variable(tf.truncated_normal([3, 3, 512, 512],
                                             dtype=tf.float32,
                                             stddev=1e-1),
                        name='weights')
    conv = tf.nn.conv2d(self.conv4_2, kernel,
                        [1, 1, 1, 1], padding='SAME')
    biases = tf.Variable(tf.constant(0.0, shape=[512],
                                     dtype=tf.float32),
                        trainable=True, name='biases')
    out = tf.nn.bias_add(conv, biases)
    self.conv4_3 = tf.nn.relu(out, name=scope)

```

```

        self.parameters += [kernel, biases]

# pool4
self.pool4 = tf.nn.max_pool(self.conv4_3,
                            ksize=[1, 2, 2, 1],
                            strides=[1, 2, 2, 1],
                            padding='SAME',
                            name='pool4')

# conv5_1
with tf.name_scope('conv5_1') as scope:
    kernel = tf.Variable(tf.truncated_normal([3, 3, 512, 512],
                                              dtype=tf.float32,
                                              stddev=1e-1),
                        name='weights')
    conv = tf.nn.conv2d(self.pool4, kernel,
                        [1, 1, 1, 1], padding='SAME')
    biases = tf.Variable(tf.constant(0.0,
                                     shape=[512], dtype=tf.float32),
                        trainable=True, name='biases')
    out = tf.nn.bias_add(conv, biases)
    self.conv5_1 = tf.nn.relu(out, name=scope)
    self.parameters += [kernel, biases]

# conv5_2
with tf.name_scope('conv5_2') as scope:
    kernel = tf.Variable(tf.truncated_normal([3, 3, 512, 512],
                                              dtype=tf.float32,
                                              stddev=1e-1),
                        name='weights')
    conv = tf.nn.conv2d(self.conv5_1, kernel,
                        [1, 1, 1, 1], padding='SAME')
    biases = tf.Variable(tf.constant(0.0,
                                     shape=[512], dtype=tf.float32),
                        trainable=True, name='biases')
    out = tf.nn.bias_add(conv, biases)
    self.conv5_2 = tf.nn.relu(out, name=scope)
    self.parameters += [kernel, biases]

# conv5_3
with tf.name_scope('conv5_3') as scope:
    kernel = tf.Variable(tf.truncated_normal([3, 3, 512, 512],
                                              dtype=tf.float32,
                                              stddev=1e-1),
                        name='weights')
    conv = tf.nn.conv2d(self.conv5_2, kernel, [1, 1, 1, 1], padding='SAME')
    biases = tf.Variable(tf.constant(0.0, shape=[512], dtype=tf.float32),
                        trainable=True, name='biases')

```

```

        out = tf.nn.bias_add(conv, biases)
        self.conv5_3 = tf.nn.relu(out, name=scope)
        self.parameters += [kernel, biases]

    # pool5
    self.pool5 = tf.nn.max_pool(self.conv5_3,
                                ksize=[1, 2, 2, 1],
                                strides=[1, 2, 2, 1],
                                padding='SAME',
                                name='pool4')

def fc_layers(self):
    # fc1
    with tf.name_scope('fc1') as scope:
        shape = int(np.prod(self.pool5.get_shape()[1:]))
        fc1w = tf.Variable(tf.truncated_normal([shape, 4096],
                                                dtype=tf.float32,
                                                stddev=1e-1),
                            name='weights')
        fc1b = tf.Variable(tf.constant(1.0, shape=[4096],
                                          dtype=tf.float32),
                            trainable=True, name='biases')
        pool5_flat = tf.reshape(self.pool5, [-1, shape])
        fc1l = tf.nn.bias_add(tf.matmul(pool5_flat, fc1w), fc1b)
        self.fc1 = tf.nn.relu(fc1l)
        self.parameters += [fc1w, fc1b]

    # fc2
    with tf.name_scope('fc2') as scope:
        fc2w = tf.Variable(tf.truncated_normal([4096, 4096],
                                                dtype=tf.float32,
                                                stddev=1e-1), name='weights')
        fc2b = tf.Variable(tf.constant(1.0, shape=[4096], dtype=tf.float32),
                            trainable=True, name='biases')
        fc2l = tf.nn.bias_add(tf.matmul(self.fc1, fc2w), fc2b)
        self.fc2 = tf.nn.relu(fc2l)
        self.parameters += [fc2w, fc2b]

    # fc3
    with tf.name_scope('fc3') as scope:
        fc3w = tf.Variable(tf.truncated_normal([4096, 1000],
                                                dtype=tf.float32,
                                                stddev=1e-1), name='weights')
        fc3b = tf.Variable(tf.constant(1.0, shape=[1000], dtype=tf.float32),
                            trainable=True, name='biases')
        self.fc3l = tf.nn.bias_add(tf.matmul(self.fc2, fc3w), fc3b)
        self.parameters += [fc3w, fc3b]

```

```

def load_weights(self, weight_file, sess):
    """
    To load the weights you need to know the order
    of the parameters as they are appended onto
    the self.parameters list
    """
    weights = np.load(weight_file)
    keys = sorted(weights.keys())
    for i, k in enumerate(keys):
        print(i, k, np.shape(weights[k]))
        sess.run(self.parameters[i].assign(weights[k]))

if __name__ == '__main__':
    sess = tf.Session()

    imgs = tf.placeholder(tf.float32, [None, 224, 224, 3])

    print('Loading model...')
    vgg = vgg16(imgs, './vggWeights/vgg16_weights.npz', sess)
    print('Done loading!')

    my_summaries = tf.summary.merge_all()
    my_writer = tf.summary.FileWriter('tb_files', sess.graph)

    img1 = imread('./vggWeights/laska.png', mode='RGB')
    img1 = imresize(img1, (224, 224))

    plt.imshow(img1)
    plt.title('Input 224x224 image')
    plt.show()

    prob, fc2_val, my_summaries_protobuf = sess.run([vgg.probs,
                                                    vgg.fc2, my_summaries],
                                                    feed_dict={vgg.imgs: [img1]})

    prob = prob[0]
    my_writer.add_summary(my_summaries_protobuf)

    num_dimensions = np.shape(fc2_val)[1]
    plt.bar(range(num_dimensions), fc2_val[0], align='center')
    plt.title('{}-dimensional representation of image'.format(num_dimensions))
    plt.show()

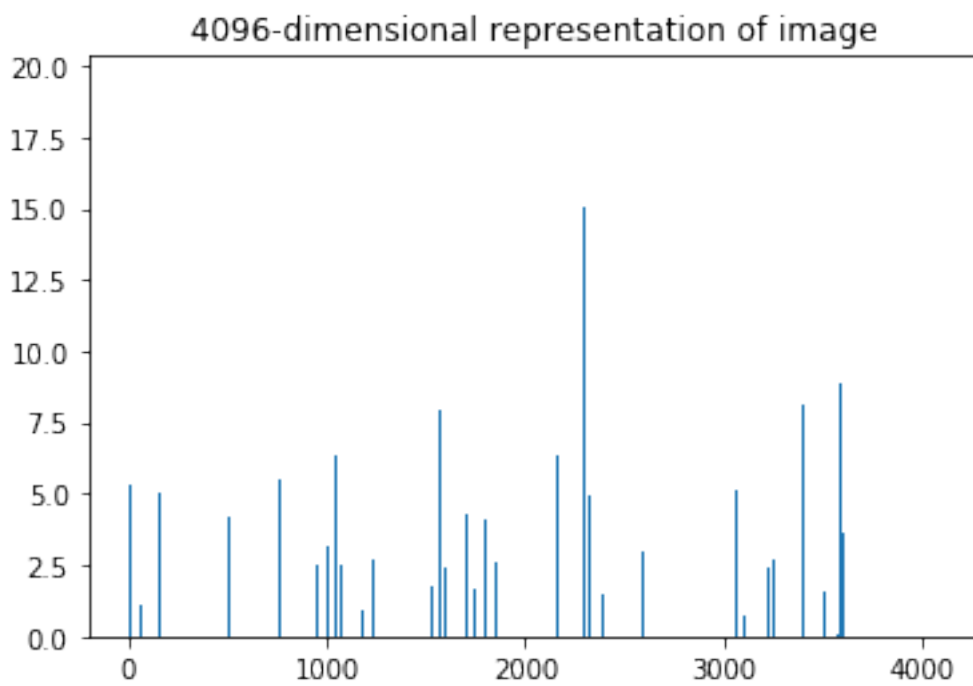
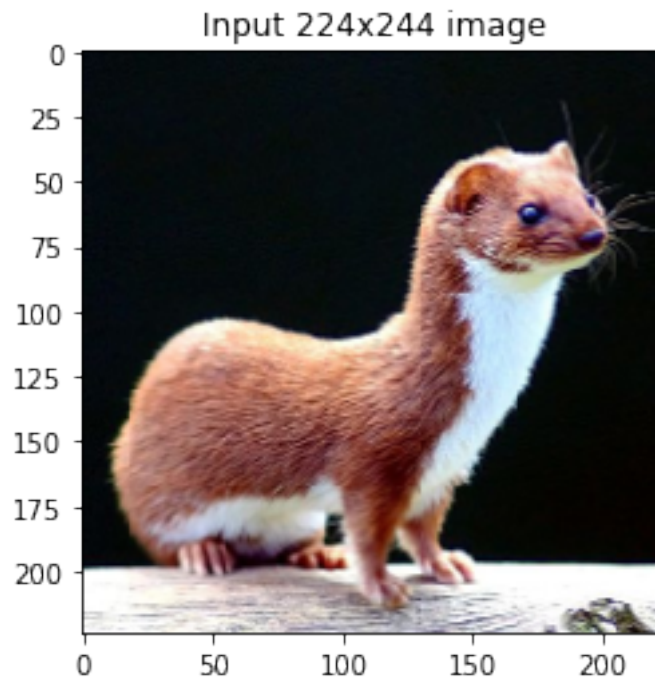
    print('Top 5 predictions of VGG-16 model:')
    preds = (np.argsort(prob)[::-1])[0:5]
    for idx, p in enumerate(preds):
        print('{} . {} ({} )'.format(idx + 1, class_names[p], prob[p]))
    sess.close()

```



```
Loading model...
0 conv1_1_W (3, 3, 3, 64)
1 conv1_1_b (64,)
2 conv1_2_W (3, 3, 64, 64)
3 conv1_2_b (64,)
4 conv2_1_W (3, 3, 64, 128)
5 conv2_1_b (128,)
6 conv2_2_W (3, 3, 128, 128)
7 conv2_2_b (128,)
8 conv3_1_W (3, 3, 128, 256)
9 conv3_1_b (256,)
10 conv3_2_W (3, 3, 256, 256)
11 conv3_2_b (256,)
12 conv3_3_W (3, 3, 256, 256)
13 conv3_3_b (256,)
14 conv4_1_W (3, 3, 256, 512)
15 conv4_1_b (512,)
16 conv4_2_W (3, 3, 512, 512)
17 conv4_2_b (512,)
18 conv4_3_W (3, 3, 512, 512)
19 conv4_3_b (512,)
20 conv5_1_W (3, 3, 512, 512)
21 conv5_1_b (512,)
22 conv5_2_W (3, 3, 512, 512)
23 conv5_2_b (512,)
24 conv5_3_W (3, 3, 512, 512)
25 conv5_3_b (512,)
26 fc6_W (25088, 4096)
27 fc6_b (4096,)
28 fc7_W (4096, 4096)
29 fc7_b (4096,)
30 fc8_W (4096, 1000)
31 fc8_b (1000,)
Done loading!
```

```
/home/damianos/.conda/envs/tfCPU/lib/python3.5/site-packages/ipykernel/__main__.py:332: Deprecat
`imread` is deprecated in SciPy 1.0.0, and will be removed in 1.2.0.
Use ``imageio.imread`` instead.
/home/damianos/.conda/envs/tfCPU/lib/python3.5/site-packages/ipykernel/__main__.py:333: Deprecat
`imresize` is deprecated in SciPy 1.0.0, and will be removed in 1.2.0.
Use ``skimage.transform.resize`` instead.
```



Top 5 predictions of VGG-16 model:  
1. weasel (0.6933858394622803)

2. polecat, fitch, foulmart, foumart, *Mustela putorius* (0.17538763582706451)
3. mink (0.12208583950996399)
4. black-footed ferret, ferret, *Mustela nigripes* (0.008870662190020084)
5. otter (0.00012108326336601749)