实验参考<http://www.cs.toronto.edu/~frossard/post/vgg16/>

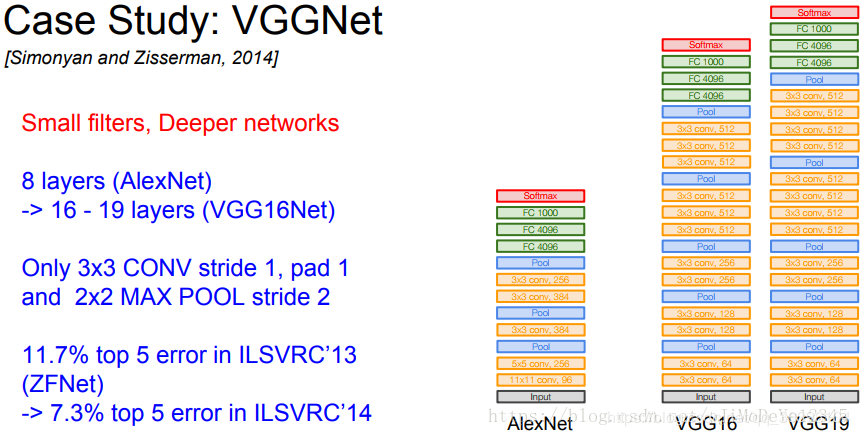
实验代码：

VGG16的实现以及原始Caffe模型的权重转换为TensorFlow

<https://github.com/ethereon/caffe-tensorflow>

[https://gist.github.com/ksimonyan/211839e770f7b538e2d8#file-readme-md](https://gist.github.com/ksimonyan/211839e770f7b538e2d8" \l "file-readme-md)

获得Npz 文件



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

self.convlayers()

self.fc\_layers()

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//预处理层使用像素值在0-255之间的RGB图像，并减去平均图像值（在整个ImageNet训练集上计算）

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.n

n.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):

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])

vgg = vgg16(imgs, 'vgg16\_weights.npz', sess)

img1 = imread('laska.png', mode='RGB')

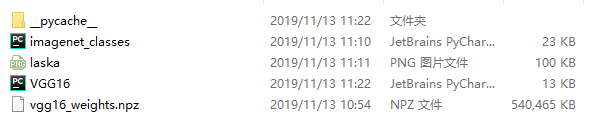
img1 = imresize(img1, (224, 224))

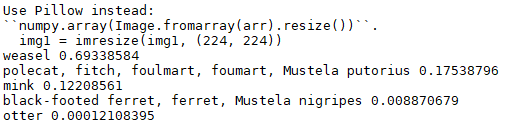
prob = sess.run(vgg.probs, feed\_dict={vgg.imgs: [img1]})[0]

preds = (np.argsort(prob)[::-1])[0:5]

for p in preds:

print (class\_names[p], prob[p])





黄鼠狼 0.69338584



