分类器训练

Tensorflow

林楚铭

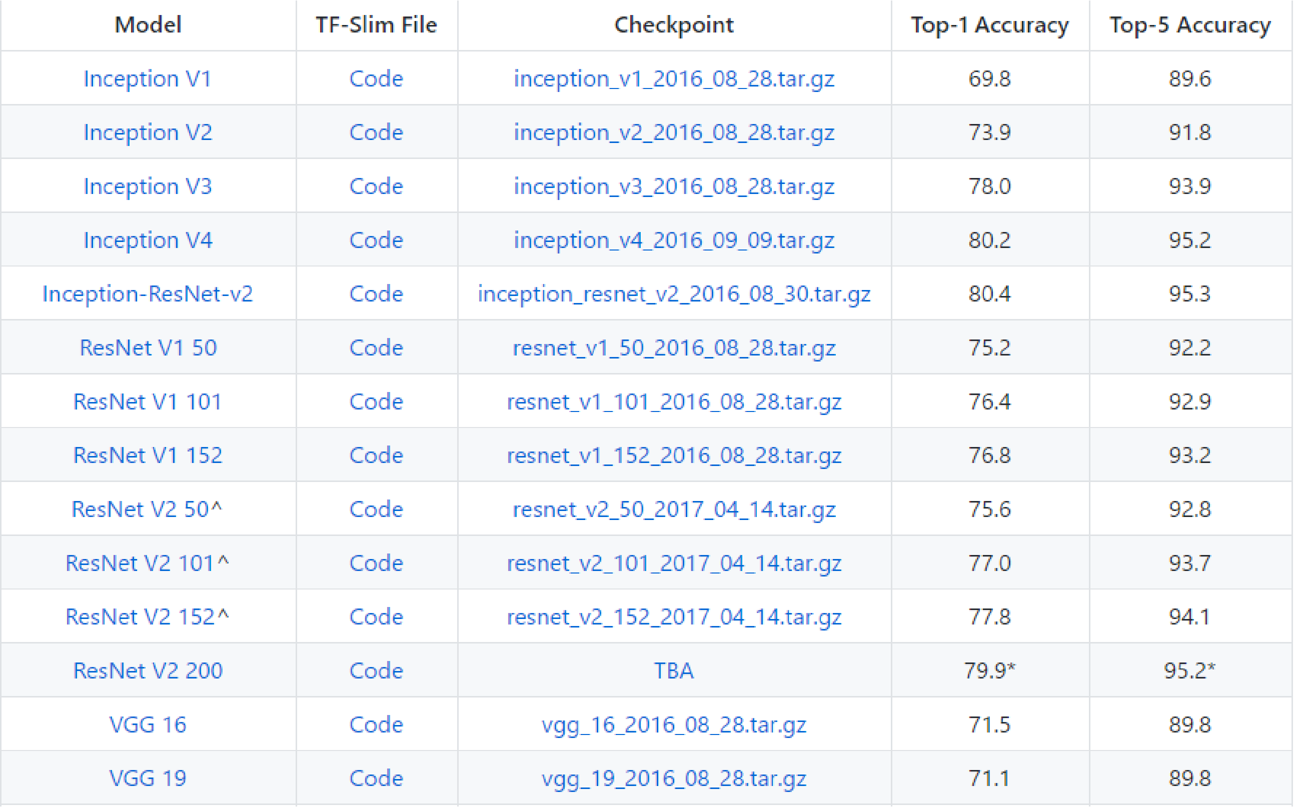
cmlin17@fudan.edu.cn

简介

- 利用在ImageNet上训练好的分类器模型，加以修改，利用fine-tune机制训练一个属于自己的分类器
- 在VOC2007数据集上训练一个1-k分类器

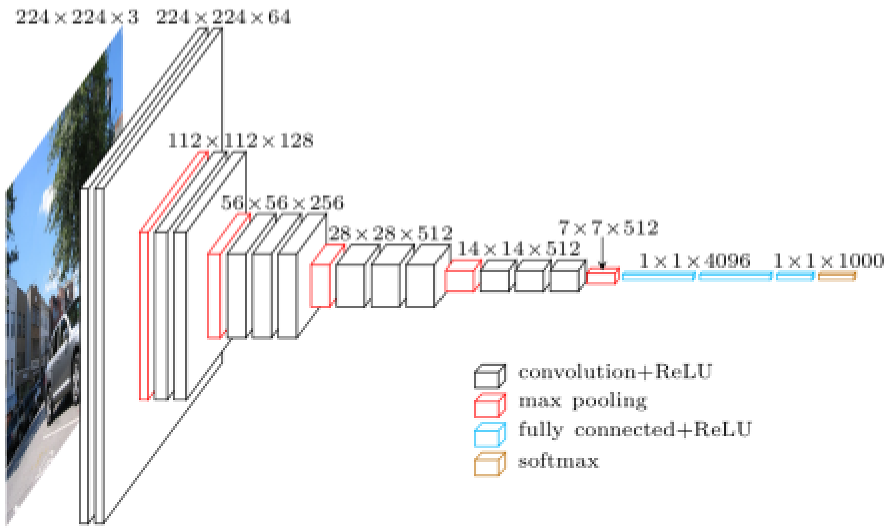
下载预训练模型

- https://github.com/tensorflow/models/tree/master/research/slim



Model	TF-Slim File	Checkpoint	Top-1 Accuracy	Top-5 Accuracy
Inception V1	Code	inception_v1_2016_08_28.tar.gz	69.8	89.6
Inception V2	Code	inception_v2_2016_08_28.tar.gz	73.9	91.8
Inception V3	Code	inception_v3_2016_08_28.tar.gz	78.0	93.9
Inception V4	Code	inception_v4_2016_09_09.tar.gz	80.2	95.2
Inception-ResNet-v2	Code	inception_resnet_v2_2016_08_30.tar.gz	80.4	95.3
ResNet V1 50	Code	resnet_v1_50_2016_08_28.tar.gz	75.2	92.2
ResNet V1 101	Code	resnet_v1_101_2016_08_28.tar.gz	76.4	92.9
ResNet V1 152	Code	resnet_v1_152_2016_08_28.tar.gz	76.8	93.2
ResNet V2 50^	Code	resnet_v2_50_2017_04_14.tar.gz	75.6	92.8
ResNet V2 101^	Code	resnet_v2_101_2017_04_14.tar.gz	77.0	93.7
ResNet V2 152^	Code	resnet_v2_152_2017_04_14.tar.gz	77.8	94.1
ResNet V2 200	Code	TBA	79.9*	95.2*
VGG 16	Code	vgg_16_2016_08_28.tar.gz	71.5	89.8
VGG 19	Code	vgg_19_2016_08_28.tar.gz	71.1	89.8

修改VGG16网络



- 训练一个新的分类器
- 去除最后一层全连接层
- 新建一层全连接层
- VOC2007一共有20分类，所以新建的全连接层的输出长度为20

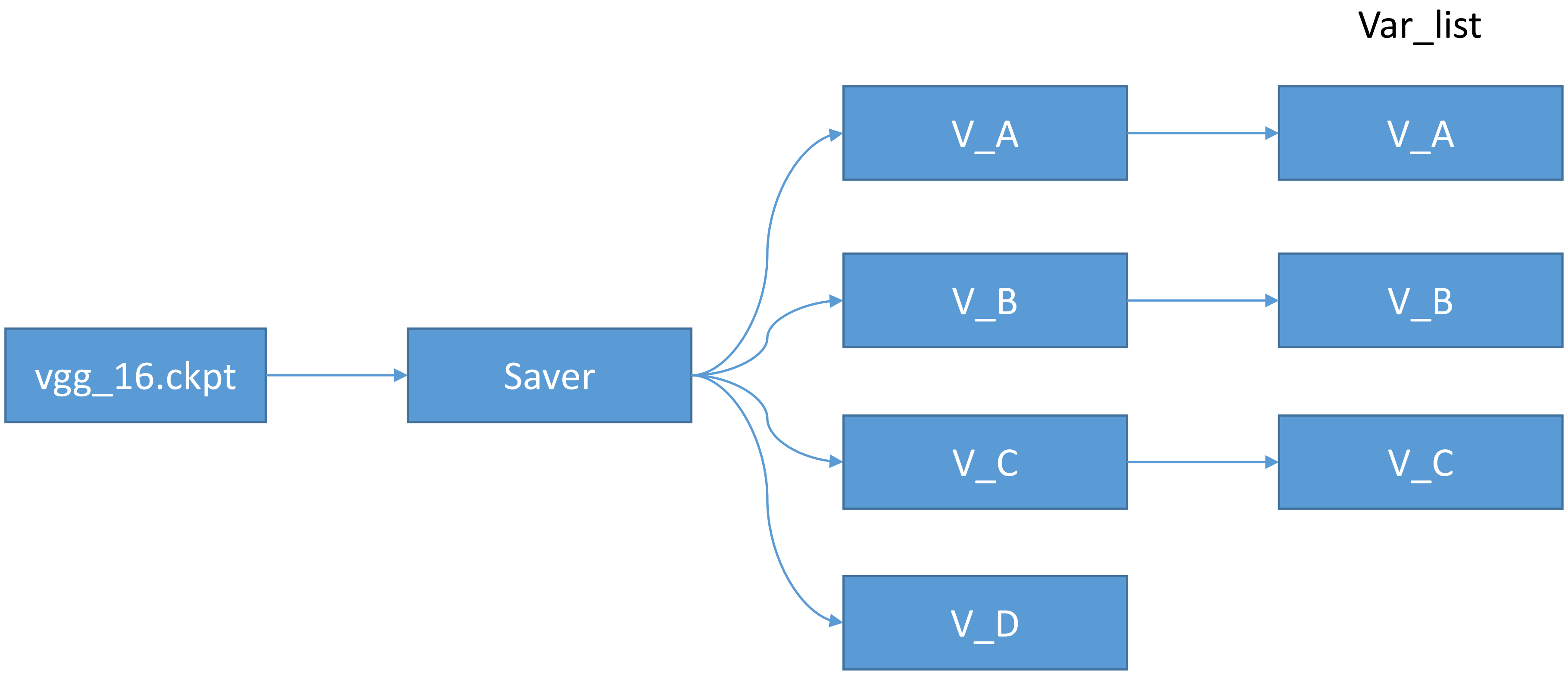
修改VGG16网络

```
if num_classes:
  net = slim.dropout(net, dropout_keep_prob, is_training=is_training,
                     scope='dropout7')
  net = slim.conv2d(net, num_classes, [1, 1],
                    activation_fn=None,
                    normalizer_fn=None,
                    scope='fc8')
```

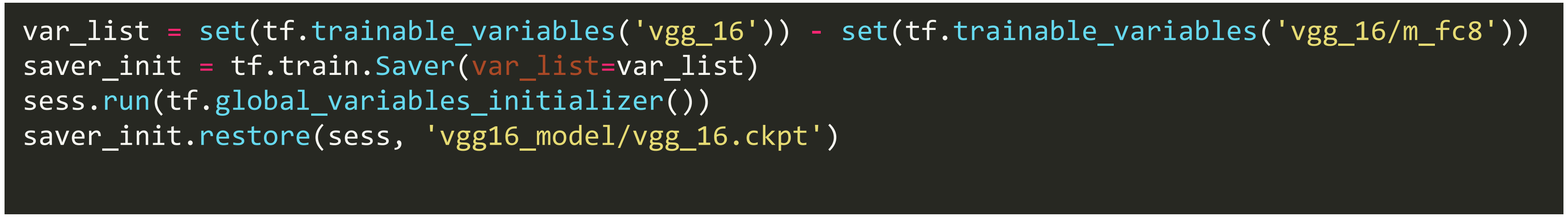
```
if num_classes:
  net = slim.dropout(net, dropout_keep_prob, is_training=is_training,
                     scope='dropout7')
  net = slim.conv2d(net, num_classes, [1, 1],
                    activation_fn=None,
                    normalizer_fn=None,
                    scope='m_fc8')
```

```
logits, end_points = our_vgg_16(img, num_classes=20)
```

读取预训练权值参数



读取预训练权值参数



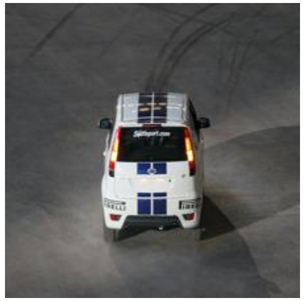
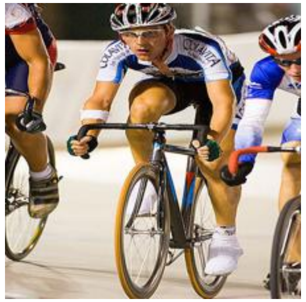
```
var_list = set(tf.trainable_variables('vgg_16')) - set(tf.trainable_variables('vgg_16/m_fc8'))
saver_init = tf.train.Saver(var_list=var_list)
sess.run(tf.global_variables_initializer())
saver_init.restore(sess, 'vgg16_model/vgg_16.ckpt')
```

数据准备1：输入图片

- 将VOC2007图片全部缩放为224x224

数据准备2： 标签

- 标签使用one hot编码

	Person	Car	Bicycle
	0	1	0
	1	0	1

K分类交叉熵损失

```
logits = tf.sigmoid(logits)
loss = -1.0 * tf.reduce_mean(gt * tf.log(logits + 1e-8) + (1 - gt) * tf.log(1 - logits + 1e-8))
```

- 使用adam优化器
- 学习率设置为1e-5
- Batch size设置为32
- 训练6000次左右即完成训练

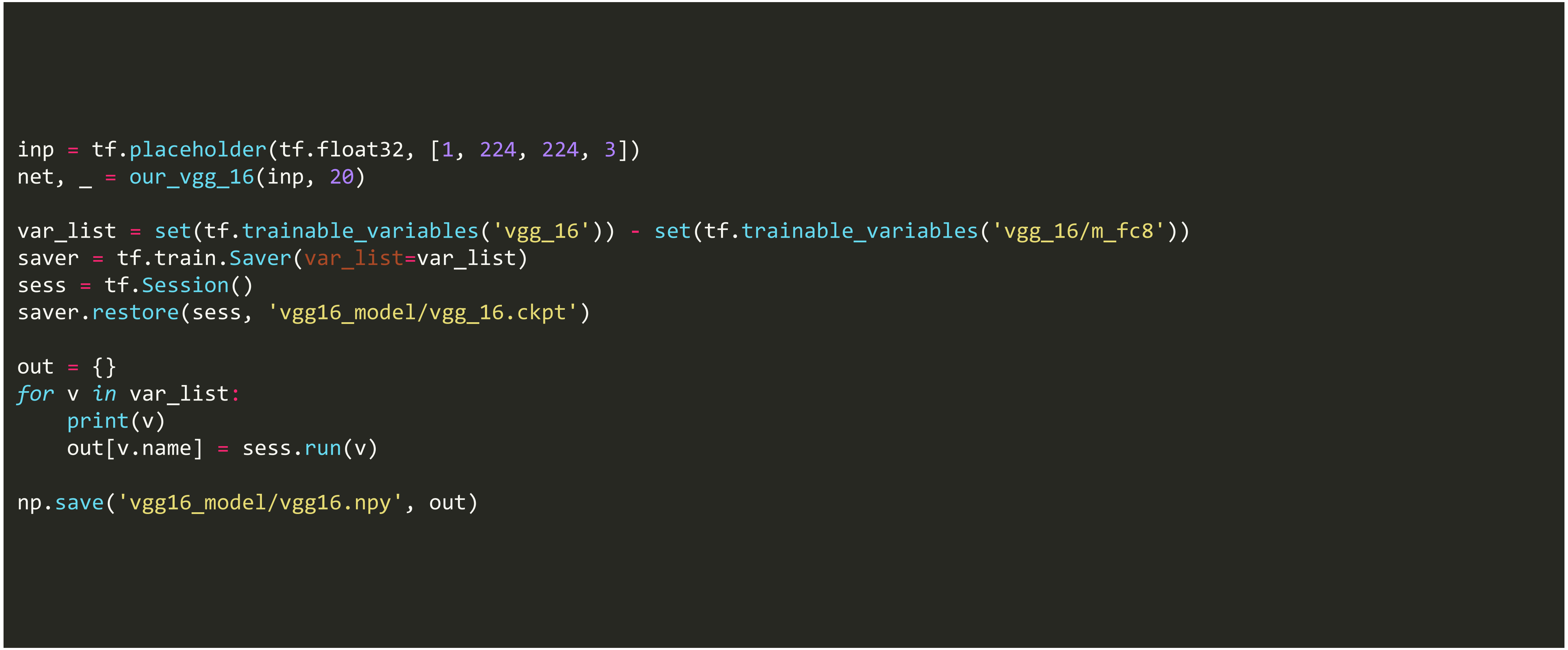
- PS：输入RGB图需减去平均值[123.68, 116.78, 103.94]

进阶

- 如果不想使用slim框架，想自己重新搭建网络
- 或者网络权值来源于其他框架，不能使用saver直接复用

- 1. 保存权值
- 2. 训练前对权值重新赋值

保存权值



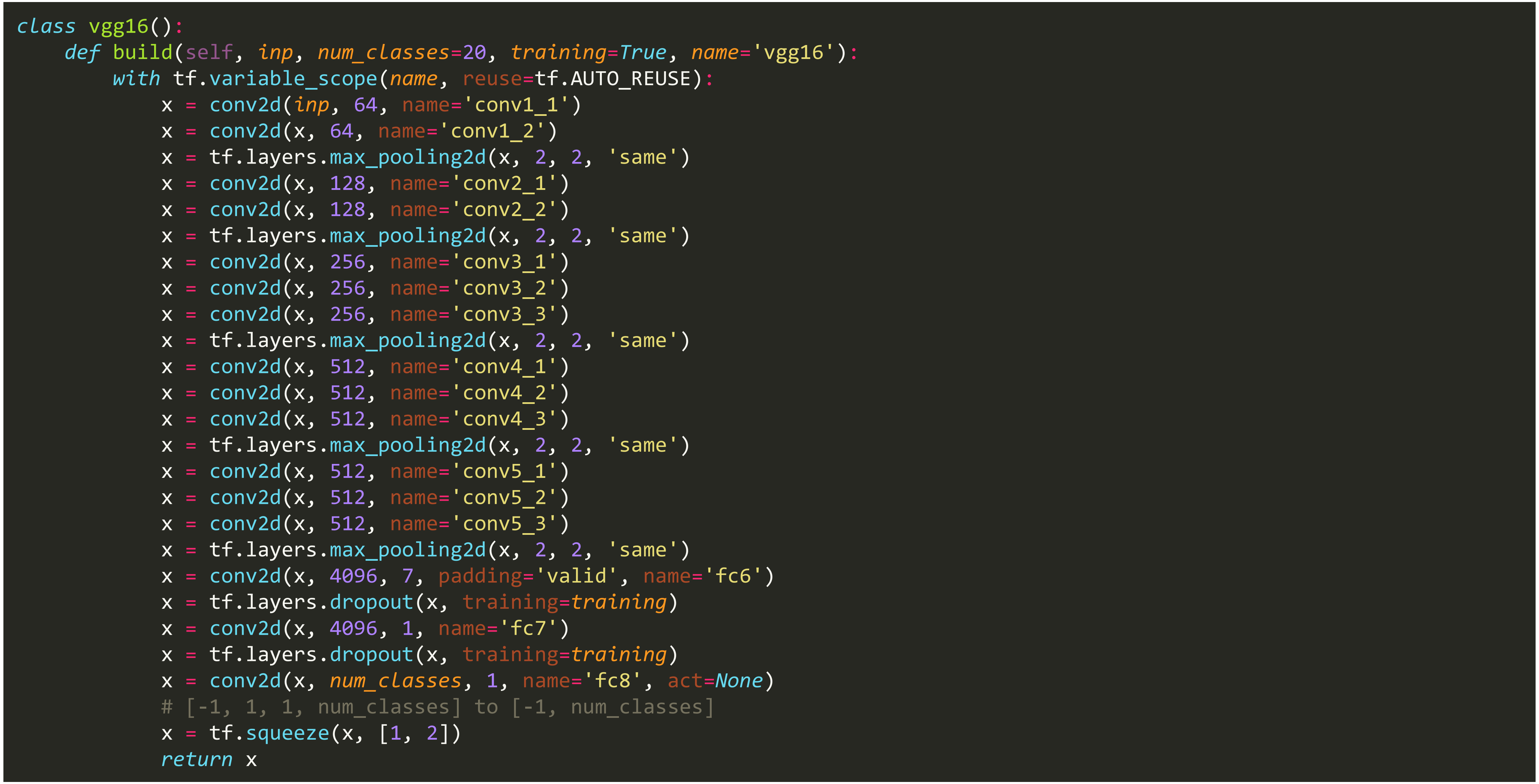
```python
inp = tf.placeholder(tf.float32, [1, 224, 224, 3])
net, _ = our_vgg_16(inp, 20)

var_list = set(tf.trainable_variables('vgg_16')) - set(tf.trainable_variables('vgg_16/m_fc8'))
saver = tf.train.Saver(var_list=var_list)
sess = tf.Session()
saver.restore(sess, 'vgg16_model/vgg_16.ckpt')

out = {}
for v in var_list:
    print(v)
    out[v.name] = sess.run(v)

np.save('vgg16_model/vgg16.npy', out)
```

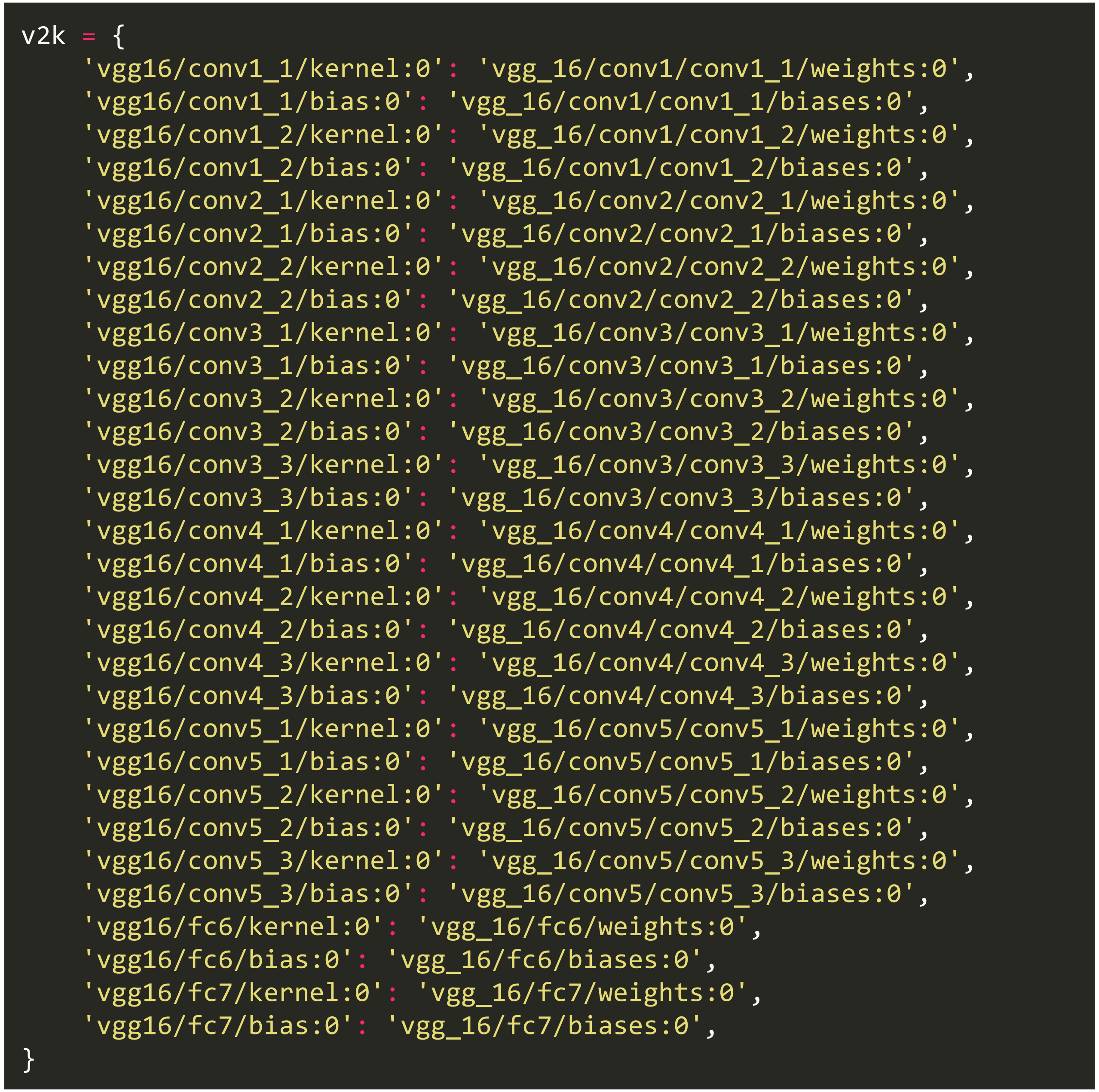
建立自己的VGG16网络



```python
class vgg16():
    def build(self, inp, num_classes=20, training=True, name='vgg16'):
        with tf.variable_scope(name, reuse=tf.AUTO_REUSE):
            x = conv2d(inp, 64, name='conv1_1')
            x = conv2d(x, 64, name='conv1_2')
            x = tf.layers.max_pooling2d(x, 2, 2, 'same')
            x = conv2d(x, 128, name='conv2_1')
            x = conv2d(x, 128, name='conv2_2')
            x = tf.layers.max_pooling2d(x, 2, 2, 'same')
            x = conv2d(x, 256, name='conv3_1')
            x = conv2d(x, 256, name='conv3_2')
            x = conv2d(x, 256, name='conv3_3')
            x = tf.layers.max_pooling2d(x, 2, 2, 'same')
            x = conv2d(x, 512, name='conv4_1')
            x = conv2d(x, 512, name='conv4_2')
            x = conv2d(x, 512, name='conv4_3')
            x = tf.layers.max_pooling2d(x, 2, 2, 'same')
            x = conv2d(x, 512, name='conv5_1')
            x = conv2d(x, 512, name='conv5_2')
            x = conv2d(x, 512, name='conv5_3')
            x = tf.layers.max_pooling2d(x, 2, 2, 'same')
            x = conv2d(x, 4096, 7, padding='valid', name='fc6')
            x = tf.layers.dropout(x, training=training)
            x = conv2d(x, 4096, 1, name='fc7')
            x = tf.layers.dropout(x, training=training)
            x = conv2d(x, num_classes, 1, name='fc8', act=None)
            # [-1, 1, 1, num_classes] to [-1, num_classes]
            x = tf.squeeze(x, [1, 2])
            return x
```

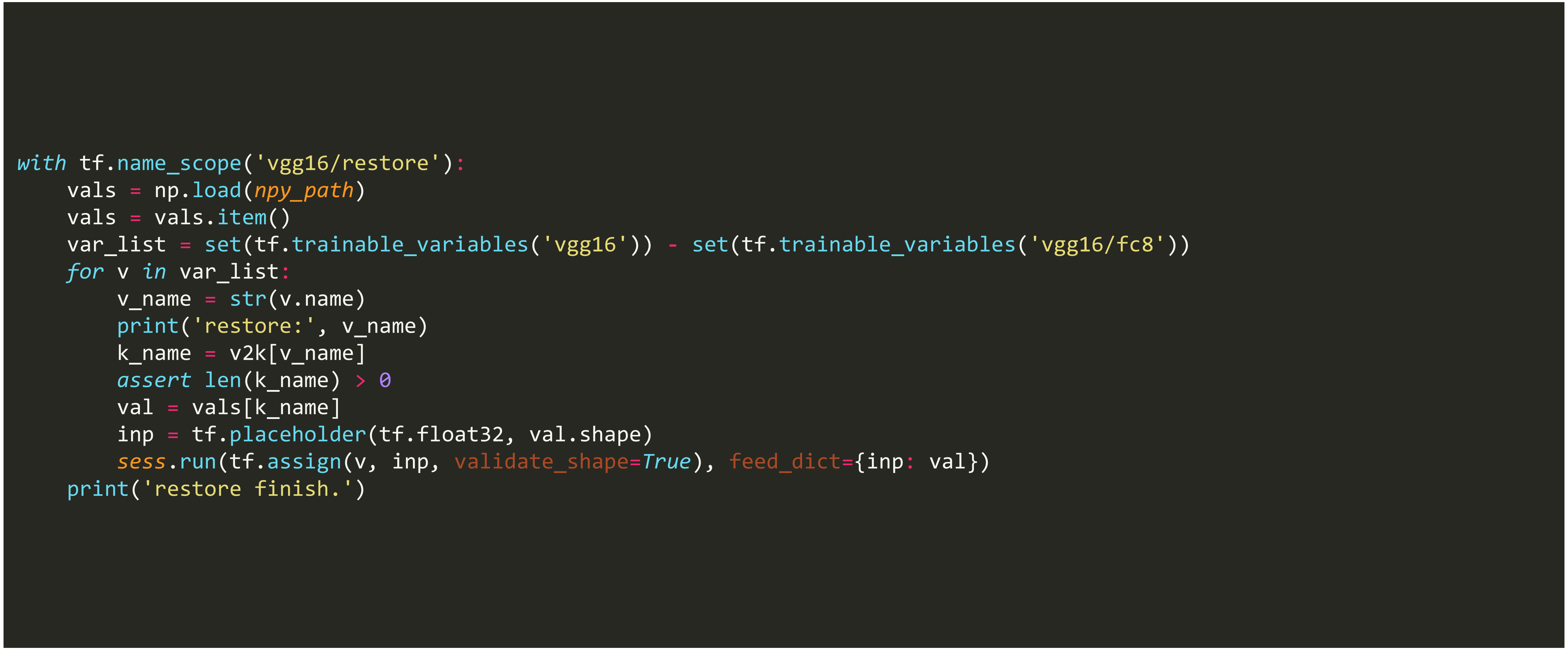
构建权值名称映射矩阵

通过使用trainable_variables获取可训练的变量，输出变量的名称，然后编写变量与变量之间对应名称的映射



```
v2k = {
    'vgg16/conv1_1/kernel:0': 'vgg_16/conv1/conv1_1/weights:0',
    'vgg16/conv1_1/bias:0': 'vgg_16/conv1/conv1_1/biases:0',
    'vgg16/conv1_2/kernel:0': 'vgg_16/conv1/conv1_2/weights:0',
    'vgg16/conv1_2/bias:0': 'vgg_16/conv1/conv1_2/biases:0',
    'vgg16/conv2_1/kernel:0': 'vgg_16/conv2/conv2_1/weights:0',
    'vgg16/conv2_1/bias:0': 'vgg_16/conv2/conv2_1/biases:0',
    'vgg16/conv2_2/kernel:0': 'vgg_16/conv2/conv2_2/weights:0',
    'vgg16/conv2_2/bias:0': 'vgg_16/conv2/conv2_2/biases:0',
    'vgg16/conv3_1/kernel:0': 'vgg_16/conv3/conv3_1/weights:0',
    'vgg16/conv3_1/bias:0': 'vgg_16/conv3/conv3_1/biases:0',
    'vgg16/conv3_2/kernel:0': 'vgg_16/conv3/conv3_2/weights:0',
    'vgg16/conv3_2/bias:0': 'vgg_16/conv3/conv3_2/biases:0',
    'vgg16/conv3_3/kernel:0': 'vgg_16/conv3/conv3_3/weights:0',
    'vgg16/conv3_3/bias:0': 'vgg_16/conv3/conv3_3/biases:0',
    'vgg16/conv4_1/kernel:0': 'vgg_16/conv4/conv4_1/weights:0',
    'vgg16/conv4_1/bias:0': 'vgg_16/conv4/conv4_1/biases:0',
    'vgg16/conv4_2/kernel:0': 'vgg_16/conv4/conv4_2/weights:0',
    'vgg16/conv4_2/bias:0': 'vgg_16/conv4/conv4_2/biases:0',
    'vgg16/conv4_3/kernel:0': 'vgg_16/conv4/conv4_3/weights:0',
    'vgg16/conv4_3/bias:0': 'vgg_16/conv4/conv4_3/biases:0',
    'vgg16/conv5_1/kernel:0': 'vgg_16/conv5/conv5_1/weights:0',
    'vgg16/conv5_1/bias:0': 'vgg_16/conv5/conv5_1/biases:0',
    'vgg16/conv5_2/kernel:0': 'vgg_16/conv5/conv5_2/weights:0',
    'vgg16/conv5_2/bias:0': 'vgg_16/conv5/conv5_2/biases:0',
    'vgg16/conv5_3/kernel:0': 'vgg_16/conv5/conv5_3/weights:0',
    'vgg16/conv5_3/bias:0': 'vgg_16/conv5/conv5_3/biases:0',
    'vgg16/fc6/kernel:0': 'vgg_16/fc6/weights:0',
    'vgg16/fc6/bias:0': 'vgg_16/fc6/biases:0',
    'vgg16/fc7/kernel:0': 'vgg_16/fc7/weights:0',
    'vgg16/fc7/bias:0': 'vgg_16/fc7/biases:0',
}
```

网络权值赋值



```python
with tf.name_scope('vgg16/restore'):
    vals = np.load(npy_path)
    vals = vals.item()
    var_list = set(tf.trainable_variables('vgg16')) - set(tf.trainable_variables('vgg16/fc8'))
    for v in var_list:
        v_name = str(v.name)
        print('restore:', v_name)
        k_name = v2k[v_name]
        assert len(k_name) > 0
        val = vals[k_name]
        inp = tf.placeholder(tf.float32, val.shape)
        sess.run(tf.assign(v, inp, validate_shape=True), feed_dict={inp: val})
    print('restore finish.')
```