VGG

VGGNet由牛津大学计算机视觉组合和Google DeepMind公司研究员一起研发的**深度卷积神经网络。**它通过反复的堆叠**3*3的小型卷积核**和**2*2的最大池化层**，成功的构建了16~19层深的卷积神经网络。VGGNet获得了ILSVRC 2014年比赛的亚军和定位项目的冠军。目前为止，VGGNet依然被用来**提取图像的特征。**

与AlexNet的区别

VGG与AlexNet在整体结构上相似，都是五个卷积层加上三个全连接层，但是他们在细节上有许多不同点。

- VGG包含**140M**个可训练参数，AlexNet只有**50M**个，所以VGG很**占内存。**
- VGG的每个卷积层都是由多个卷积层构成的，可以称之为**卷积块**，而AlexNet的卷积层就是单个的。
- VGG从始至终使用的都是**3*3的卷积核**，而AlexNet并不是。
- VGG的最大池化层是**2*2的且步长为2**，不是重叠层，而AlexNet的最大池化层是3*3的且步长为2，是重叠层。
- VGG的每个卷积块后都会跟上一个最大池化层，而AlexNet只有第一个和第二个和第五个卷积层后面跟最大池化层。
- VGG抛弃了AlexNet中的LRN，因为LRN会**降低学习速度**并且还没有明显的效果，AlexNet中只有第一层和第二层使用LRN。
- 相比于AlexNet，VGG的网络模型更加**清晰规范。**

整体架构

ConvNet Configuration					
A	A-LRN	B	C	D	E
11 weight layers	11 weight layers	13 weight layers	16 weight layers	16 weight layers	19 weight layers
input (224 × 224 RGB image)					
conv3-64	conv3-64 **LRN**	conv3-64 **conv3-64**	conv3-64 conv3-64	conv3-64 conv3-64	conv3-64 conv3-64
maxpool					
conv3-128	conv3-128	conv3-128 **conv3-128**	conv3-128 conv3-128	conv3-128 conv3-128	conv3-128 conv3-128
maxpool					
conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256 **conv1-256**	conv3-256 conv3-256 **conv3-256**	conv3-256 conv3-256 conv3-256 **conv3-256**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
FC-4096					
FC-4096					
FC-1000					
soft-max					

由此可见，VGG有多种架构，其中网络D是VGG-16，E是VGG-19。并且通道数都是从**64成倍递增至256**。

3*3卷积核

VGG从始至终用的都是3*3卷积核，并用很容易知道连续2个3*3卷积核构成的卷积块相当于5*5的卷积核，连续3个3*3的卷积核构成的卷积块相当于7*7的卷积核。那为什么VGG不用单个的5*5或7*7卷积核呢？论文中解释的原因有：

1. 采用3个而不是1个非线性校正层，使**决策函数具有更强的分辨力。**
2. **减少参数的数量**，假设输入输出都是C通道，那么3个3*3卷积核总共的参数为3*（3*3*C）=27C个，而1个7*7的卷积核有7*7*C=49C个参数，**多了81%的参数数量**，这可以看作是对7*7的卷积核进行了**正则化。**

1*1卷积核

我们在网络C中可以看见第三四五个卷积块**最后跟了1个1*1的卷积核**，它是在**不影响卷积层感受野**的情况下增加**决策函数非线性**的方法。

初始化

首先训练相对较浅的网络A，用**随机初始化**训练。然后训练后面更深的网络时，用网络A的前四个卷积层和最后三个全连接层初始化，中间的层使用随机初始化。随机初始化在**N(0,**10^{-2}**)**中取样，偏差全部初始化为0。

超参数

VGG的超参数设置和AlexNet大体相似。

- mini-batch=256,动量=0.9
- 权重衰减=$5*10^{-4}$,Dropout参数为0.5。
- 学习速率初始化为10^{-2},当验证集的准确性停止提高时降低10倍。
- epochs=74。

代码实现

```
import tensorflow as tf

def vgg_block(filters,channels):
    vb = tf.keras.Sequential()
    for _ in range(filters):
        vb.add(tf.keras.layers.Conv2D(channels,(3,3),padding='same',activation='relu'))
    vb.add(tf.keras.layers.MaxPool2D(2,2))
    return vb

def vgg(conv_arch):
    net = tf.keras.Sequential()
    for (filters,channels) in conv_arch:
        net.add(vgg_block(filters,channels))
    net.add(tf.keras.Sequential([
        tf.keras.layers.Flatten(),
        tf.keras.layers.Dense(4096,activation='relu'),
        tf.keras.layers.Dropout(0.5),
        tf.keras.layers.Dense(4096,activation='relu'),
        tf.keras.layers.Dropout(0.5),
        tf.keras.layers.Dense(1000,activation='softmax')
    ]))
    return net

#VGG-19
conv_arch = ((2,64),(2,128),(4,256),(4,512),(4,512))

net = vgg(conv_arch)
```

模型参数

```
Model: "sequential_7"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
sequential_8 (Sequential)    (1, 112, 112, 64)         38720
_________________________________________________________________
sequential_9 (Sequential)    (1, 56, 56, 128)          221440
_________________________________________________________________
sequential_10 (Sequential)   (1, 28, 28, 256)          2065408
_________________________________________________________________
sequential_11 (Sequential)   (1, 14, 14, 512)          8259584
_________________________________________________________________
sequential_12 (Sequential)   (1, 7, 7, 512)            9439232
_________________________________________________________________
sequential_13 (Sequential)   (1, 1000)                 123642856
=================================================================
Total params: 143,667,240
Trainable params: 143,667,240
Non-trainable params: 0
```

备注

我的代码只是实现了VGG大体的架构，对于权重初始化是tensorflow中默认的方式，并不是原论文中的方式。