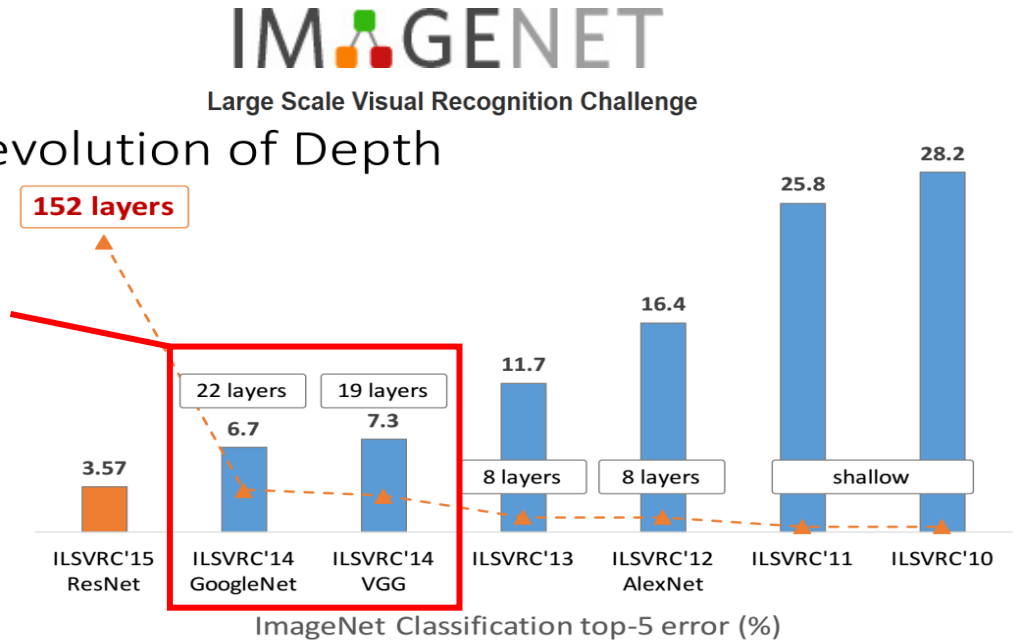
九曲阑干

AI深度学习

VGG论文解读

VGG背景介绍

2014年ImageNet比赛
分类任务
冠军GoogLeNet
亚军VGG



标题

发表会议：ICLR2015
（该会议专注于深度学习和强化学习，水平与NeurIPS，ICML相当）

Published as a conference paper at ICLR 2015

VERY DEEP CONVOLUTIONAL NETWORKS FOR LARGE-SCALE IMAGE RECOGNITION

标题：很深的卷积神经网络
（16层和19层）

Karen Simonyan* **& Andrew Zisserman+**
Visual Geometry Group, Department of Engineering Science, University of Oxford
{karen,az}@robots.ox.ac.uk

作者：牛津大学VGG团队，计算机视觉老牌大牛团队
（后来两位作者都加入DeepMind）

论文目录

九曲阑干

Abstract

1 Introduction

2 ConvNet Configurations
- 2.1 Architecture
- 2.2 Configurations
- 2.3 Discussion

3 Classification Framework
- 3.1 Training
- 3.2 Testing
- 3.3 Implementation Details

4 Classification Experiments
- 4.1 Single scale Evaluation
- 4.2 Multi-scale Evaluation
- 4.3 Multi-crop Evaluation
- 4.4 Model Fusion
- 4.5 Comparison with SOTA

5 Conclusion

列举论文要点

论文总体介绍

论文所提方法的详细介绍

实验结果和分析

论文总结

注意：这篇论文写作不太像一般论文
非常像实验报告
有很多实验设置可以参考

Introduction

Introduction概述了论文的主要内容和贡献

1 INTRODUCTION

With ConvNets becoming more of a commodity in the computer vision field, a number of attempts have been made to improve the original architecture of Krizhevsky et al. (2012) in a bid to achieve better accuracy. For instance, the best-performing submissions to the ILSVRC-2013 (Zeiler & Fergus, 2013; Sermanet et al., 2014) utilised smaller receptive window size and smaller stride of the first convolutional layer. Another line of improvements dealt with training and testing the networks densely over the whole image and over multiple scales (Sermanet et al., 2014; Howard, 2014). In this paper, we address another important aspect of ConvNet architecture design – its depth. To this end, we fix other parameters of the architecture, and steadily increase the depth of the network by adding more convolutional layers, which is feasible due to the use of very small (3×3) convolution filters in all layers.

As a result, we come up with significantly more accurate ConvNet architectures, which not only achieve the state-of-the-art accuracy on ILSVRC classification and localisation tasks, but are also applicable to other image recognition datasets, where they achieve excellent performance even when used as a part of a relatively simple pipelines (e.g. deep features classified by a linear SVM without fine-tuning). We have released our two best-performing models[1] to facilitate further research.

论文的目标：改进AlexNet的网络结构，提升ImageNet图像分类的精度

论文怎么做的：增加卷积层的层数，增加网络深度

取得的效果
①在ILSVRC分类和定位任务中都获得优异的结果
②可以泛化到其他分类数据集上（做预训练模型）

总结VGG的贡献：
结构简洁，堆叠3x3卷积组成卷积块（启发了后面的网络设计，如resnet）
在其他数据集的泛化性好，用于提取图像特征（如应用于风格迁移）

网络结构

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					

①
②
③
④
⑤

VGG只使用3x3卷积
5个卷积块 + 3个全连接层

卷积块由3x3卷积(pad=1)组成，
每个卷积层保持输出大小=输入大小

卷积块内部没有pooling，
每个卷积块后面有max-pool
max-pool的池化核为2，stride=2，
max-pool后特征图长宽缩小一半

卷积块的通道数从64开始逐渐翻倍，最后到512

每个卷积层后面都使用ReLU激活函数
去掉了LRN（因为对精度提升没有帮助）

全连接层设置和AlexNet相同

网络结构

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**
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**
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**
maxpool					
FC-4096					
FC-4096					
FC-1000					
soft-max					

① ② ③ ④ ⑤

输入224x224x3

第一卷积块，输出224x224x64
第一maxpool，输出112x112x64

第二卷积块，输出112x112x128
第二maxpool，输出56x56x128

第三卷积块，输出56x56x256
第三maxpool，输出28x28x256

第四卷积块，输出28x28x512
第四maxpool，输出14x14x512

第五卷积块，输出14x14x512
第五maxpool，输出7x7x512

第一全连接，4096
第二全连接，4096
第三全连接，1000

网络结构

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					

① 从A到E共5种结构

② 每种结构的不同在于卷积块中包含的卷积层个数不同

③ 两种后来常用的网络结构：
D为VGG16
E为VGG19

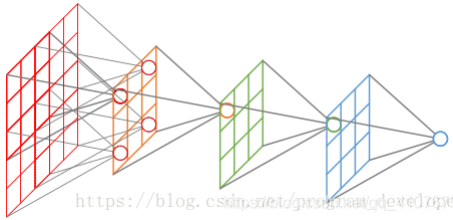
④ （如风格迁移中使用VGG19提取特征）

⑤ 19层精度已经基本饱和，再增加层数也不会提升结果，所以没有再设计更深的网络

网络结构

2.3 Discussion

Our ConvNet configurations are quite different from the ones used in the top-performing entries of the ILSVRC-2012 (Krizhevsky et al., 2012) and ILSVRC-2013 competitions (Zeiler & Fergus, 2013; Sermanet et al., 2014). Rather than using relatively large receptive fields in the first conv. layers (e.g. 11×11 with stride 4 in (Krizhevsky et al., 2012), or 7×7 with stride 2 in (Zeiler & Fergus, 2013; Sermanet et al., 2014)), we use very small 3×3 receptive fields throughout the whole net, which are convolved with the input at every pixel (with stride 1). It is easy to see that a stack of two 3×3 conv. layers (without spatial pooling in between) has an effective receptive field of 5×5; three such layers have a 7×7 effective receptive field. So what have we gained by using, for instance, a stack of three 3×3 conv. layers instead of a single 7×7 layer? First, we incorporate three non-linear rectification layers instead of a single one, which makes the decision function more discriminative. Second, we decrease the number of parameters: assuming that both the input and the output of a three-layer 3×3 convolution stack has C channels, the stack is parametrised by $3\left(3^2C^2\right) = 27C^2$ weights; at the same time, a single 7×7 conv. layer would require $7^2C^2 = 49C^2$ parameters, i.e. 81% more. This can be seen as imposing a regularisation on the 7×7 conv. filters, forcing them to have a decomposition through the 3×3 filters (with non-linearity injected in between).



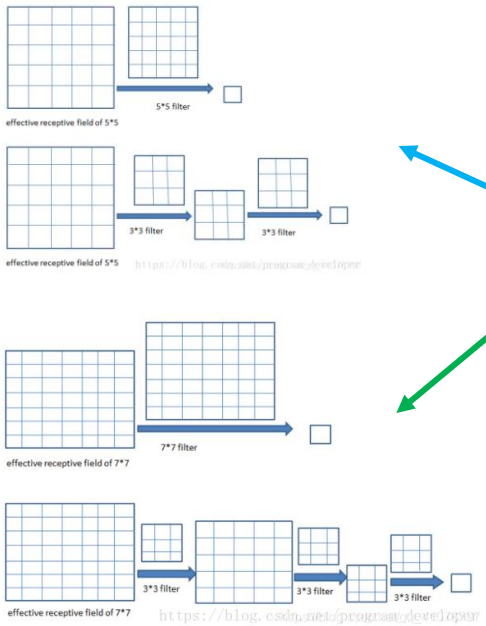
如何理解卷积块?

文中描述:
VGG在整个网络中使用了非常小的3x3卷积
堆叠两个3x3卷积的感受野等于一个5x5卷积
堆叠三个3x3卷积的感受野等于一个7x7卷积

感受野：计算卷积时，输出特征图的像素点在输入特征图上映射的区域，即输入特征图一个点对应输入特征图上的区域

神经科学角度理解：感受野是神经网络中，一个神经元“看到”的区域

网络结构



如何理解卷积块?

卷积块可以理解为，用多个3x3卷积去实现相同感受野的一个卷积核较大的卷积层

堆叠两个3x3卷积的感受野等于一个5x5卷积
堆叠三个3x3卷积的感受野等于一个7x7卷积

感受野：计算卷积时，输出特征图的像素点在输入特征图上映射的区域，即输入特征图一个点对应输入特征图上的区域

神经科学角度理解：感受野是神经网络中，一个神经元“看到”的区域

网络结构

2.3 DISCUSSION

Our ConvNet configurations are quite different from the ones used in the top-performing entries of the ILSVRC-2012 (Krizhevsky et al., 2012) and ILSVRC-2013 competitions (Zeiler & Fergus, 2013; Sermanet et al., 2014). Rather than using relatively large receptive fields in the first conv. layers (e.g. 11×11 with stride 4 in (Krizhevsky et al., 2012), or 7×7 with stride 2 in (Zeiler & Fergus, 2013; Sermanet et al., 2014)), we use very small 3×3 receptive fields throughout the whole net, which are convolved with the input at every pixel (with stride 1). It is easy to see that a stack of two 3×3 conv. layers (without spatial pooling in between) has an effective receptive field of 5×5; three such layers have a 7×7 effective receptive field. So what have we gained by using, for instance, a stack of three 3×3 conv. layers instead of a single 7×7 layer? First, we incorporate three non-linear rectification layers instead of a single one, which makes the decision function more discriminative. Second, we decrease the number of parameters: assuming that both the input and the output of a three-layer 3×3 convolution stack has C channels, the stack is parametrised by $3\left(3^2C^2\right) = 27C^2$ weights; at the same time, a single 7×7 conv. layer would require $7^2C^2 = 49C^2$ parameters, i.e. 81% more. This can be seen as imposing a regularisation on the 7×7 conv. filters, forcing them to have a decomposition through the 3×3 filters (with non-linearity injected in between).

如何理解卷积块？

卷积块可以理解为，用多个3x3卷积去实现相同感受野的一个卷积核较大的卷积层

堆叠两个3x3卷积的感受野等于一个5x5卷积
堆叠三个3x3卷积的感受野等于一个7x7卷积

使用多个3x3代替一个5x5或7x7的好处：
①两个3x3中间有ReLU，增加了非线性操作，所以可以增加判别能力
②使用多个3x3可以减少参数个数（可以缓解过拟合）

多个3x3减少参数个数，举例：
假如通道数是C，用三个3x3卷积代替一个7x7卷积
三个3x3卷积的参数量=$3(3^2C^2)=27C^2$
一个7x7卷积的参数量=$49C^2$
一个7x7卷积比三个3x3卷积的参数量多81%

训练设置

3.1 TRAINING

The ConvNet training procedure generally follows Krizhevsky et al. (2012) (except for sampling the input crops from multi-scale training images, as explained later). Namely, the training is carried out by optimising the multinomial logistic regression objective using mini-batch gradient descent (based on back-propagation (LeCun et al., 1989)) with momentum. The batch size was set to 256, momentum to 0.9. The training was regularised by weight decay (the L_2 penalty multiplier set to $5 \cdot 10^{-4}$) and dropout regularisation for the first two fully-connected layers (dropout ratio set to 0.5). The learning rate was initially set to 10^{-2}, and then decreased by a factor of 10 when the validation set accuracy stopped improving. In total, the learning rate was decreased 3 times, and the learning was stopped after 370K iterations (74 epochs). We conjecture that in spite of the larger number of parameters and the greater depth of our nets compared to (Krizhevsky et al., 2012), the nets required less epochs to converge due to (a) implicit regularisation imposed by greater depth and smaller conv. filter sizes; (b) pre-initialisation of certain layers.

The initialisation of the network weights is important, since bad initialisation can stall learning due to the instability of gradient in deep nets. To circumvent this problem, we began with training the configuration A (Table 1), shallow enough to be trained with random initialisation. Then, when training deeper architectures, we initialised the first four convolutional layers and the last three fully-connected layers with the layers of net A (the intermediate layers were initialised randomly). We did not decrease the learning rate for the pre-initialised layers, allowing them to change during learning. For random initialisation (where applicable), we sampled the weights from a normal distribution with the zero mean and 10^{-2} variance. The biases were initialised with zero. It is worth noting that after the paper submission we found that it is possible to initialise the weights without pre-training by using the random initialisation procedure of Glorot & Bengio (2010).

一些重要的训练超参数：
batchsize=256
momentum=0.9（动量）
weight decay=5x10^{-4}（L2正则化）
dropout=0.5
学习率：初始0.01，当验证集精度停止提升时乘0.1，共下降三次
训练次数：74epochs

训练设置

momentum动量法

思路：mini-batch计算的梯度有一定的偏差，会导致训练线路轨迹不断抖动，训练非常慢
做法：对每次迭代计算的梯度进行加权平均
作用：抵消梯度的偏差，缓解训练轨迹的抖动，从而加快训练速度

(a) SGD without momentum

原始（无动量）

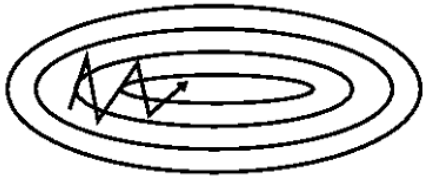
$$W_t \leftarrow W_{t-1} - \eta g_t$$

动量法

动量 动量系数 学习率 梯度

$$v_t \leftarrow \beta v_{t-1} + \eta g_t$$

$$W_t \leftarrow W_{t-1} - v_t$$

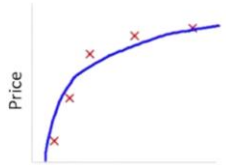


(b) SGD with momentum

训练设置

weight decay权重衰减

weight decay：对权重进行L2正则化
作用：约束权重weight的复杂程度
效果：缓解过拟合



$\theta_0 + \theta_1 x + \theta_2 x^2$

加正则化

$\theta_0 + \theta_1 x + \theta_2 x^2 + \theta_3 x^3 + \theta_4 x^4$

不加正则化

目标函数 $L_{wd} = L + \frac{\lambda}{2}||W||^2$

L2正则化项

梯度更新 $W_t \leftarrow W_{t-1} - \eta(g_t + \lambda W_{t-1})$

学习率　梯度　L2正则化系数
即weight decay系数

训练设置

3.1 TRAINING

The ConvNet training procedure generally follows Krizhevsky et al. (2012) (except for sampling the input crops from multi-scale training images, as explained later). Namely, the training is carried out by optimising the multinomial logistic regression objective using mini-batch gradient descent (based on back-propagation (LeCun et al., 1989)) with momentum. The batch size was set to 256, momentum to 0.9. The training was regularised by weight decay (the L_2 penalty multiplier set to $5 \cdot 10^{-4}$) and dropout regularisation for the first two fully-connected layers (dropout ratio set to 0.5). The learning rate was initially set to 10^{-2}, and then decreased by a factor of 10 when the validation set accuracy stopped improving. In total, the learning rate was decreased 3 times, and the learning was stopped after 370K iterations (74 epochs). We conjecture that in spite of the larger number of parameters and the greater depth of our nets compared to (Krizhevsky et al., 2012), the nets required less epochs to converge due to (a) implicit regularisation imposed by greater depth and smaller conv. filter sizes; (b) pre-initialisation of certain layers.

The initialisation of the network weights is important, since bad initialisation can stall learning due to the instability of gradient in deep nets. To circumvent this problem, we began with training the configuration A (Table 1), shallow enough to be trained with random initialisation. Then, when training deeper architectures, we initialised the first four convolutional layers and the last three fully-connected layers with the layers of net A (the intermediate layers were initialised randomly). We did not decrease the learning rate for the pre-initialised layers, allowing them to change during learning. For random initialisation (where applicable), we sampled the weights from a normal distribution with the zero mean and 10^{-2} variance. The biases were initialised with zero. It is worth noting that after the paper submission we found that it is possible to initialise the weights without pre-training by using the random initialisation procedure of Glorot & Bengio (2010).

网络初始化：
最初的做法：先训练最浅的网络A，训练更深的网络时，前四个卷积层和后三个全连接层用模型A的参数初始化

后来作者发现用Xavier方法可以对更深的网络也直接用随机数初始化

训练设置

Xavier初始化

来源于《Understanding the difficulty of training deep feedforward neural networks》，以一作Xavier的名字命名

基本思路：为了防止出现梯度消失或梯度爆炸，合理的初始化应当保证让输入和输出的分布相同
由于初始化通常采用均匀分布或高斯分布（均值为0），需计算合理的初始化分布标准差

从前向传播推导，方差应保证 $\forall i,\ n_i Var[W^i] = 1$

从反向传播推导，方差应保证 $\forall i,\ n_{i+1} Var[W^i] = 1$

折中后

均匀分布： $W^i \sim U(-a, a),\ a = \beta \times \sqrt{\frac{6}{n_i + n_{i+1}}}$

高斯分布： $W^i \sim N(0, \sigma^2),\ \sigma = \beta \times \sqrt{\frac{2}{n_i + n_{i+1}}}$

PyTorch实现：
torch.nn.init.xavier_uniform_(weight, gain=1.0)
torch.nn.init.xavier_normal_(weight, gain=1.0)

公式中β是可调的缩放系数，即代码中gain

训练设置

Kaiming初始化（对Xavier初始化的改进）

来源于《 Delving deep into rectifiers: Surpassing human-level performance on imagenet classification 》，以一作何恺明的名字命名（ResNet使用了这种初始化方法）

基本思路：保证让输入和输出的分布相同，计算合理的初始化分布标准差
Xavier初始化中仅考虑了sigmoid和tanh，Kaiming初始化考虑了ReLU

从前向传播推导，方差应保证 $\frac{1}{2}n_l Var[w_l] = 1, \ \forall l$
从反向传播推导，方差应保证 $\frac{1}{2}n_{l+1} Var[w_l] = 1, \ \forall l$

因为ReLU将一半神经元置为0，公式前面多了一个1/2

高斯分布初始化 $W_l \sim N(0, \sqrt{2/n_l})$

PyTorch实现：
torch.nn.init. kaiming_normal_(weight, mode='fan_in')

训练数据扩增

We consider two approaches for setting the training scale S. The first is to fix S, which corresponds to single-scale training (note that image content within the sampled crops can still represent multi-scale image statistics). In our experiments, we evaluated models trained at two fixed scales: $S = 256$ (which has been widely used in the prior art (Krizhevsky et al., 2012; Zeiler & Fergus, 2013; Sermanet et al., 2014)) and $S = 384$. Given a ConvNet configuration, we first trained the network using $S = 256$. To speed-up training of the $S = 384$ network, it was initialised with the weights pre-trained with $S = 256$, and we used a smaller initial learning rate of 10^{-3}.

The second approach to setting S is multi-scale training, where each training image is individually rescaled by randomly sampling S from a certain range $[S_{min}, S_{max}]$ (we used $S_{min} = 256$ and $S_{max} = 512$). Since objects in images can be of different size, it is beneficial to take this into account during training. This can also be seen as training set augmentation by scale jittering, where a single

数据扩增：同AlexNet，首先使用了水平翻转和颜色扩增，然后：

单尺度训练：使用256和384两种尺度，从中随机取224大小的图像快
（相当于固定图片尺度，增加图像平移扩增）

多尺度训练：使用256~512之间任意尺度对图像缩放，再随机取224大小的图像块
（相当于在图像平移扩增同时，加入图像缩放扩增）

测试

3.2 TESTING

At test time, given a trained ConvNet and an input image, it is classified in the following way. First, it is isotropically rescaled to a pre-defined smallest image side, denoted as Q (we also refer to it as the test scale). We note that Q is not necessarily equal to the training scale S (as we will show in Sect. 4, using several values of Q for each S leads to improved performance). Then, the network is applied densely over the rescaled test image in a way similar to (Sermanet et al., 2014). Namely, the fully-connected layers are first converted to convolutional layers (the first FC layer to a 7×7 conv. layer, the last two FC layers to 1×1 conv. layers). The resulting fully-convolutional net is then applied to the whole (uncropped) image. The result is a class score map with the number of channels equal to the number of classes, and a variable spatial resolution, dependent on the input image size. Finally, to obtain a fixed-size vector of class scores for the image, the class score map is spatially averaged (sum-pooled). We also augment the test set by horizontal flipping of the images; the soft-max class posteriors of the original and flipped images are averaged to obtain the final scores for the image.

Since the fully-convolutional network is applied over the whole image, there is no need to sample multiple crops at test time (Krizhevsky et al., 2012), which is less efficient as it requires network re-computation for each crop. At the same time, using a large set of crops, as done by Szegedy et al. (2014), can lead to improved accuracy, as it results in a finer sampling of the input image compared to the fully-convolutional net. Also, multi-crop evaluation is complementary to dense evaluation due to different convolution boundary conditions: when applying a ConvNet to a crop, the convolved feature maps are padded with zeros, while in the case of dense evaluation the padding for the same crop naturally comes from the neighbouring parts of an image (due to both the convolutions and spatial pooling), which substantially increases the overall network receptive field, so more context is captured. While we believe that in practice the increased computation time of multiple crops does not justify the potential gains in accuracy, for reference we also evaluate our networks using 50 crops per scale (5×5 regular grid with 2 flips), for a total of 150 crops over 3 scales, which is comparable to 144 crops over 4 scales used by Szegedy et al. (2014).

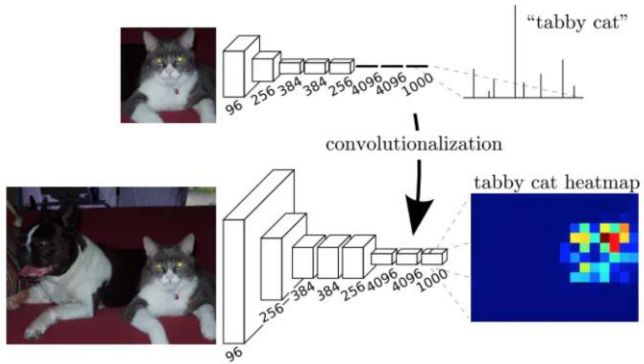
Dense Evaluation方法：
将第一个全连接层变成7x7卷积层，
后两个全连接层变成1x1卷积层，
最后将结果的softmax概率图取平均

这样计算效率较高（只需要跑一次模型，而multi-crop需要每个crop跑一次，跑很多次模型）

这样输入图像大于224，也可以计算
（这个思路就是全卷积网络，广泛应用于图像分割）

Dense Evaluation和取多个crop再平均的区别：
结果比较相近
主要差异来源于：
multi-crop中每个crop都是用0padding
dense evaluation的每个crop相当于是用邻域的图像作padding

测试



Dense Evaluation方法：
将第一个全连接层变成7x7卷积层，
后两个全连接层变成1x1卷积层，
最后将结果的softmax概率图取平均

这样计算效率较高（只需要跑一次模型，而multi-crop需要每个crop跑一次，跑很多次模型）

这样输入图像大于224，也可以计算
（这个思路就是全卷积网络，广泛应用于图像分割）

Dense Evaluation和取多个crop再平均的区别：
结果比较相近
主要差异来源于：
multi-crop中每个crop都是用0padding
dense evaluation的每个crop相当于是用邻域的图像作padding

GPU训练

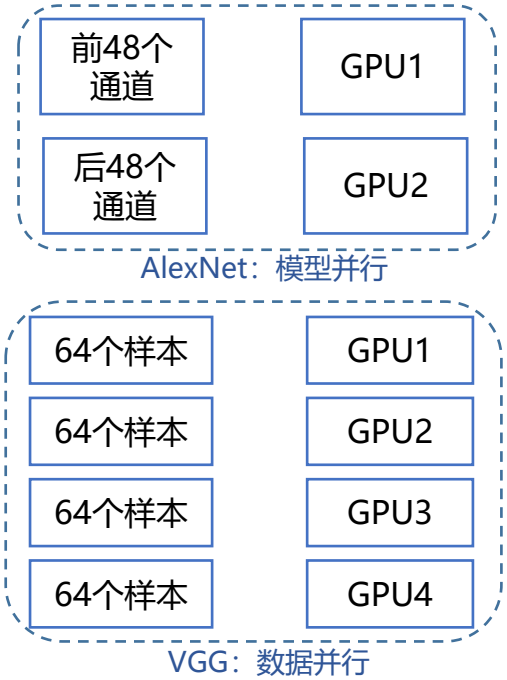
3.3 Implementation Details

Our implementation is derived from the publicly available C++ Caffe toolbox (Jia, 2013) (branched out in December 2013), but contains a number of significant modifications, allowing us to perform training and evaluation on multiple GPUs installed in a single system, as well as train and evaluate on full-size (uncropped) images at multiple scales (as described above). Multi-GPU training exploits data parallelism, and is carried out by splitting each batch of training images into several GPU batches, processed in parallel on each GPU. After the GPU batch gradients are computed, they are averaged to obtain the gradient of the full batch. Gradient computation is synchronous across the GPUs, so the result is exactly the same as when training on a single GPU.

While more sophisticated methods of speeding up ConvNet training have been recently proposed (Krizhevsky, 2014), which employ model and data parallelism for different layers of the net, we have found that our conceptually much simpler scheme already provides a speedup of 3.75 times on an off-the-shelf 4-GPU system, as compared to using a single GPU. On a system equipped with four NVIDIA Titan Black GPUs, training a single net took 2–3 weeks depending on the architecture.

GPU训练

使用了数据并行
每个batch中的样本平均分布在各个GPU上，计算出梯度后，再把整个batch的梯度平均（梯度同步更新）
4块GPU，加速比达到3.75



实验

Table 3: **ConvNet performance at a single test scale.**

ConvNet config. (Table 1)	smallest image side		top-1 val. error (%)	top-5 val. error (%)
	train (S)	test (Q)		
A	256	256	29.6	10.4
A-LRN	256	256	29.7	10.5
B	256	256	28.7	9.9
C	256	256	28.1	9.4
	384	384	28.1	9.3
	[256;512]	384	27.3	8.8
D	256	256	27.0	8.8
	384	384	26.8	8.7
	[256;512]	384	25.6	8.1
E	256	256	27.3	9.0
	384	384	26.9	8.7
	[256;512]	384	**25.5**	**8.0**

LRN对结果提升帮助不大，因此去掉LRN

多尺度训练对结果提升较为明显

随着网络深度增加，精度不断提升
到19层后精度趋于饱和（19层和16层结果差不多）
因此VGG没有再设计更深的网络

使用层数深、卷积核小的网络，优于使用层数浅、卷积核大的网络

实验

Table 4: **ConvNet performance at multiple test scales.**

ConvNet config. (Table 1)	smallest image side		top-1 val. error (%)	top-5 val. error (%)
	train (S)	test (Q)		
B	256	224,256,288	28.2	9.6
C	256	224,256,288	27.7	9.2
	384	352,384,416	27.8	9.2
	[256; 512]	256,384,512	26.3	8.2
D	256	224,256,288	26.6	8.6
	384	352,384,416	26.5	8.6
	[256; 512]	256,384,512	**24.8**	**7.5**
E	256	224,256,288	26.9	8.7
	384	352,384,416	26.7	8.6
	[256; 512]	256,384,512	**24.8**	**7.5**

在测试时也使用多尺度，可以进一步提升精度

ConvNet config. (Table 1)	Evaluation method	top-1 val. error (%)	top-5 val. error (%)
D	dense	24.8	7.5
	multi-crop	24.6	7.5
	multi-crop & dense	**24.4**	**7.2**
E	dense	24.8	7.5
	multi-crop	24.6	7.4
	multi-crop & dense	**24.4**	**7.1**

使用multi-crop evaluation和dense evaluation测试结果差不多，且二者可以互补，共同使用可以进一步提升测试结果

实验

Table 6: **Multiple ConvNet fusion results.**

Combined ConvNet models	Error		
	top-1 val	top-5 val	top-5 test
ILSVRC submission			
(D/256/224,256,288), (D/384/352,384,416), (D/[256;512]/256,384,512) (C/256/224,256,288), (C/384/352,384,416) (E/256/224,256,288), (E/384/352,384,416)	24.7	7.5	7.3
post-submission			
(D/[256;512]/256,384,512), (E/[256;512]/256,384,512), dense eval.	24.0	7.1	7.0
(D/[256;512]/256,384,512), (E/[256;512]/256,384,512), multi-crop	23.9	7.2	-
(D/[256;512]/256,384,512), (E/[256;512]/256,384,512), multi-crop & dense eval.	**23.7**	**6.8**	**6.8**

训练多个模型，把softmax概率平均，可以进一步提升测试结果

实验

Table 11: **Comparison with the state of the art in image classification on VOC-2007, VOC-2012, Caltech-101, and Caltech-256**. Our models are denoted as "VGG". Results marked with * were achieved using ConvNets pre-trained on the *extended* ILSVRC dataset (2000 classes).

Method	VOC-2007 (mean AP)	VOC-2012 (mean AP)	Caltech-101 (mean class recall)	Caltech-256 (mean class recall)
Zeiler & Fergus (Zeiler & Fergus, 2013)	-	79.0	86.5 ± 0.5	74.2 ± 0.3
Chatfield et al. (Chatfield et al., 2014)	82.4	83.2	88.4 ± 0.6	77.6 ± 0.1
He et al. (He et al., 2014)	82.4	-	**93.4 ± 0.5**	-
Wei et al. (Wei et al., 2014)	81.5 (85.2*)	81.7 (**90.3***)	-	-
VGG Net-D (16 layers)	89.3	89.0	91.8 ± 1.0	85.0 ± 0.2
VGG Net-E (19 layers)	89.3	89.0	92.3 ± 0.5	85.1 ± 0.3
VGG Net-D & Net-E	**89.7**	**89.3**	92.7 ± 0.5	**86.2 ± 0.3**

（来源于附录）
VGG具有很好的泛化性
将VGG模型在ImageNet预训练后，再微调其他数据集，获得了比其他网络更优的结果

论文要点整理

九曲阑干

- VGG的网络结构有什么特点？卷积层和maxpool是按什么规律排布的？
- 如何理解VGG中的卷积块？多个3x3为什么可以代替一个大卷积核？代替后有哪些好处？
- VGG中是否使用了LRN，为什么？
- 什么是感受野？
- 训练中使用动量，会对梯度产生什么影响？使用动量有什么好处？
- weight decay对目标函数有什么影响？使用weight decay有什么好处？
- VGG是如何初始化的？
- 如何理解Xavier初始化和Kaiming初始化的设计思路？
- VGG在训练和测试时都使用了多尺度图像扩增，有什么效果？
- VGG还尝试了多个模型融合，有什么效果？
- 为什么VGG最深为19层，不再继续增加层数？