目 录

写在前面

Life, thin and light-off time and time again
Frivolous tireless
生命，一次又一次轻薄过
轻狂不知疲倦

ImageNet Classification with Deep Convolutional Neural Networks

基于深度卷积神经网络的图像分类网络

0.1 Abstract(摘要)

We trained a large, deep convolutional neural network to classify the 1.2 million high-resolution images in the ImageNet LSVRC-2010 contest into the 1000 different classes. On the test data, we achieved top-1 and top-5 error rates of 37.5% and 17.0% which is considerably better than the previous state-of-the-art. The neural network, which has 60 million parameters and 650,000 neurons, consists of five convolutional layers, some of which are followed by max-pooling layers, and three fully-connected layers with a final 1000-way softmax. To make training faster, we used non-saturating neurons and a very efficient GPU implementation of the convolution operation. To reduce overfitting in the fully-connected layers we employed a recently-developed regularization method called “dropout” that proved to be very effective. We also entered a variant of this model in the ILSVRC-2012 competition and achieved a winning top-5 test error rate of 15.3%, compared to 26.2% achieved by the second-best entry.

在 ImageNet LSVRC-2010 比赛中，我们训练了一个大型深度卷积神经网络，将 120 万张高分辨率图像分为 1000 个不同的种类。在测试集上，我们得到了 top-1 37.5%, top-5 17.0% 的错误率，这比目前最先进的结果还要好得多。这个神经网络有 6000 万个参数和 650000 个神经元，包含 5 个卷积层（某些卷积层后带有池化层）和 3 个全连接层，最后是一个 1000 维的 softmax。为了训练更快，我们使用了非饱和神经元，并且针对卷积操作进行了非常有效的 GPU 实现。为了减少全连接层的过拟合，我们采用了最近新开发的名为 “dropout” 的正则化方法，并且结果证明是非常有效的，我们同样使用这个模型的变种模型，参加 ILSVRC-2012 竞赛，以 top-5 15.3% 的错误率赢得了冠军，而第二名错误率是 26.2%.

0.2 Introduction(引言)

Current approaches to object recognition make essential use of machine learning methods. To improve their performance, we can collect larger datasets, learn more powerful models, and use better techniques for preventing overfitting. Until recently, datasets of labeled images were relatively small — on the order of tens of thousands of images (e.g., NORB [16], Caltech-101/256 [8, 9], and CIFAR-10/100 [12]). Simple recognition tasks can be solved quite well with datasets of this size, especially if they are augmented with label-preserving transformations. For example, the currentbest error rate on the MNIST digit-recognition task (<0.3%) approaches human performance [4]. But objects in realistic settings exhibit consid-

erable variability, so to learn to recognize them it is necessary to use much larger training sets. And indeed, the shortcomings of small image datasets have been widely recognized (e.g., Pinto et al. [21]), but it has only recently become possible to collect labeled datasets with millions of images. The new larger datasets include LabelMe [23], which consists of hundreds of thousands of fully-segmented images, and ImageNet [6], which consists of over 15 million labeled high-resolution images in over 22,000 categories.

目前目标识别不可避免的使用机器学习的方法。为了提高它们的效率，我们通过收集更多的数据，学习更加强大的模型，使用更好的技术来防止过拟合。直到最近为止，被标准的图像数据集仍相对较小-在数万张图像的量级上（例如，NORB[16]，Caltech-101/256 [8, 9] 和 CIFAR-10/100 [12]）。简单的识别任务在这样大小的数据集上可以被解决的相当好，特别是当它们通过保留标签的转换进行扩充时。例如，目前在 MNIST 数字识别任务上最好的识别准确率已经接近人类水平（<0.3）。但是，模型在真实环境下展现出相当大的不定性，因此为了学习识别它们，有必要使用更大的训练数据集。实际上，小图像数据集的缺点已经被广泛认识到（例如，Pinto et al. [21]），但是收集上百万图像的标注数据仅在最近才变得的可行。新的更大的数据集包括 LabelMe [23]，它包含了数十万张完全分割的图像，ImageNet[6]，它包含了 22000 个类别上的超过 1500 万张标注的高分辨率的图像。

To learn about thousands of objects from millions of images, we need a model with a large learning capacity. However, the immense complexity of the object recognition task means that this problem cannot be specified even by a dataset as large as ImageNet, so our model should also have lots of prior knowledge to compensate for all the data we don't have. Convolutional neural networks (CNNs) constitute one such class of models [16, 11, 13, 18, 15, 22, 26]. Their capacity can be controlled by varying their depth and breadth, and they also make strong and mostly correct assumptions about the nature of images (namely, stationarity of statistics and locality of pixel dependencies). Thus, compared to standard feedforward neural networks with similarly-sized layers, CNNs have much fewer connections and parameters and so they are easier to train, while their theoretically-best performance is likely to be only slightly worse.

为了从数百万张图像中学习几千个对象，我们需要一个有很强学习能力的模型。然而对象识别任务的巨大复杂性意味着这个问题不能被指定，即使通过像 ImageNet 这样的大数据集，因此我们的模型应该也有许多先验知识来补偿我们所没有的数据。卷积神经网络 (CNNs) 构成了一个这样的模型 [16, 11, 13, 18, 15, 22, 26]。它们的能力可以通过改变它们的广度和深度来控制，它们也可以对图像的本质进行强大且通常正确的假设（也就是说，统计的稳定性和像素依赖的局部性）。因此，与具有层次大小相似的标准前馈神经网络，CNNs 有更少的连接和参数，因此它们更容易训练，而它们理论上的最佳性能可能仅比标准前馈神经网络差一点。

Despite the attractive qualities of CNNs, and despite the relative efficiency of their local architecture, they have still been prohibitively expensive to apply in large scale to high-resolution images. Luckily, current GPUs, paired with a highly-optimized implementation of 2D convolution, are powerful enough to facilitate the training of interestingly-large CNNs, and recent datasets such as ImageNet contain enough labeled examples to train such models without severe overfitting.

尽管 CNN 具有引人注目的质量，尽管它们的局部架构相当有效，但将它们大规模的应用到到高分辨率图像中仍然是极其昂贵的。幸运的是，目前的 GPU，搭配了高度优化的 2D 卷积实现，强大到足够促进有趣地大量 CNN 的训练，最近的数据集例如 ImageNet 包含足够的标注样本来训练这样的模型而没有严重的过拟合。

The specific contributions of this paper are as follows: we trained one of the largest convolutional neural networks to date on the subsets of ImageNet used in the ILSVRC-2010 and ILSVRC-2012 competitions [2] and achieved by far the best results ever reported on these datasets. We wrote a highly-optimized GPU implementation of 2D convolution and all the other operations inherent in training convolutional neural networks, which we make available publicly. Our network contains a number of new and unusual features which improve its performance and reduce its training time, which are detailed in Section 3. The size of our network made overfitting a significant problem, even with 1.2 million labeled training examples, so we used several effective techniques for preventing overfitting, which are described in Section 4. Our final network contains five convolutional and three fully-connected layers, and this depth seems to be important: we found that removing any convolutional layer (each of which contains no more than 1% of the model's parameters) resulted in inferior performance.

本文具体的贡献如下：我们在 ILSVRC-2010 和 ILSVRC-2012[2] 的 ImageNet 子集上训练了到目前为止最大的神经网络之一，并取得了迄今为止在这些数据集上报道过的最好结果。我们编写了高度优化的 2D 卷积 GPU 实现以及训练卷积神经网络内部的所有其它操作，我们把它公开了。我们的网络包含许多新的不寻常的特性，这些特性提高了神经网络的性能并减少了训练时间，详见第三节。即使使用了 120 万标注的训练样本，我们的网络尺寸仍然使过拟合成为一个明显的问题，因此我们使用了一些有效的技术来防止过拟合，详见第四节。我们最终的网络包含 5 个卷积层和 3 个全连接层，深度似乎是非常重要的：我们发现移除任何卷积层（每个卷积层包含的参数不超过模型参数的 1%）都会导致更差的性能。

In the end, the network's size is limited mainly by the amount of memory available on current GPUs and by the amount of training time that we are willing to tolerate. Our network takes between five and six days to train on two GTX 580 3GB GPUs. All of our experiments suggest that our results can be improved simply by waiting for faster GPUs and bigger datasets to become available.

最后，网络尺寸主要受限于目前 GPU 的内存容量和我们能忍受的训练时间。我们的网络在两个 GTX 580 3GB GPU 上训练五六天。我们的所有实验表明我们的结果可以简单地通过等待更快的 GPU 和更大的可用数据集来提高。

0.3 The Dataset（数据集）

ImageNet is a dataset of over 15 million labeled high-resolution images belonging to roughly 22,000 categories. The images were collected from the web and labeled by human labelers using Amazon's Mechanical Turk crowd-sourcing tool. Starting in 2010, as part of the Pascal Visual Object Challenge, an annual competition called the ImageNet Large-Scale Visual Recognition Challenge (ILSVRC) has been held. ILSVRC uses a subset of ImageNet with roughly 1000 images in each of 1000 categories. In all, there are roughly 1.2 million training images, 50,000 validation images, and 150,000 testing images.

ImageNet 数据集有超过 1500 万的标注高分辨率图像，这些图像属于大约 22000 个类别。这些图像是从网上收集的，使用了 Amazon's Mechanical Turk 的众包工具通过人工标注的。从 2010 年起，作为 Pascal 视觉对象挑战赛的一部分，每年都会举办 ImageNet 大规模视觉识别挑战赛（ILSVRC）。ILSVRC 使用 ImageNet 的一个子集，1000 个类别每个类别大约 1000 张图像。总计，大约 120 万训练图像，50000 张验证图像和 15 万测试图像。

ILSVRC-2010 is the only version of ILSVRC for which the test set labels are available, so this is the version on which we performed most of our experiments. Since we also entered our model in the ILSVRC-2012 competition, in Section 6 we report our results on this version of the dataset as well, for which test set labels are unavailable. On ImageNet, it is customary to report two error rates: top-1 and top-5, where the top-5 error rate is the fraction of test images for which the correct label is not among the five labels considered most probable by the model.

ILSVRC-2010 是 ILSVRC 竞赛中唯一可以获得测试集标签的版本，因此我们大多数实验都是在这个版本上运行的。由于我们也使用我们的模型参加了 ILSVRC-2012 竞赛，因此在第六节我们也报告了模型在这个版本的数据集上的结果，这个版本的测试标签是不可获得的。在 ImageNet 上，按照惯例报告两个错误率：top-1 和 top-5，top-5 错误率是指测试图像的正确标签不在模型认为的五个最可能的便签之中。

ImageNet consists of variable-resolution images, while our system requires a constant input dimensionality. Therefore, we down-sampled the images to a fixed resolution of 256 × 256. Given a rectangular image, we first rescaled the image such that the shorter side was of length 256, and then cropped out the central 256×256 patch from the resulting image. We did not pre-process the images in any other way, except for subtracting the mean activity over the training set from each pixel. So we trained our network on the (centered) raw RGB values of

the pixels.

ImageNet 包含各种分辨率的图像，而我们的系统要求不变的输入维度。因此，我们将图像进行下采样到固定的 256×256 分辨率。给定一个矩形图像，我们首先缩放图像短边长度为 256，然后从结果图像中裁剪中心的 256×256 大小的图像块。除了在训练集上对像素减去平均活跃度外，我们不对图像做任何其它的预处理。因此我们在原始的 RGB 像素值（中心的）上训练我们的网络。

0.4 The Architecture（架构）

The architecture of our network is summarized in Figure 2. It contains eight learned layers — five convolutional and three fully-connected. Below, we describe some of the novel or unusual features of our network's architecture. Sections 3.1-3.4 are sorted according to our estimation of their importance, with the most important first.

我们的网络架构概括为图 2。它包含八个学习层–5 个卷积层和 3 个全连接层。下面，我们将描述我们网络结构中的一些新奇的不寻常的特性。3.1-3.4 小节按照我们对它们评估的重要性进行排序，最重要的最优先。

0.4.1 ReLU Nonlinearity（ReLU 非线性）

tandard way to model a neuron's output f as a function of its input x is with $f(x) = tanh(x)$ or $f(x) = (1 + e^x)^1$. In terms of training time with gradient descent, these saturating nonlinearities are much slower than the non-saturating nonlinearity $f(x) = max(0, x)$. Following Nair and Hinton [20], we refer to neurons with this nonlinearity as Rectified Linear Units (ReLUs). Deep convolutional neural networks with ReLUs train several times faster than their equivalents with tanh units. This is demonstrated in Figure 1, which shows the number of iterations required to reach 25% training error on the CIFAR-10 dataset for a particular four-layer convolutional network. This plot shows that we would not have been able to experiment with such large neural networks for this work if we had used traditional saturating neuron models.

将神经元输出 f 建模为输入 x 的函数的标准方式是用 $f(x) = tanh(x)$ 或 $f(x) = (1 + e^x)^1$ 。考虑到梯度下降的训练时间，这些饱和的非线性比非饱和非线性 $f(x) = max(0, x)$ 更慢。根据 Nair 和 Hinton[20] 的说法，我们将这种非线性神经元称为修正线性单元 (ReLU)。采用 ReLU 的深度卷积神经网络训练时间比等价的 $tanh(x)$ 单元要快几倍。在图 1 中，对于一个特定的四层卷积网络，在 CIFAR-10 数据集上达到 25% 的训练误差所需要的迭代次数可以证实这一点。这幅图表明，如果我们采用传统的饱和神经元模型，我们将不能在如此大的神经网络上实验该工作。

We are not the first to consider alternatives to traditional neuron models in CNNs. For example, Jarrett et al. [11] claim that the nonlinearity $f(x) = |tanh(x)|$ works particularly well

with their type of contrast normalization followed by local average pooling on the Caltech-101 dataset. However, on this dataset the primary concern is preventing overfitting, so the effect they are observing is different from the accelerated ability to fit the training set which we report when using ReLUs. Faster learning has a great influence on the performance of large models trained on large datasets.

我们不是第一个考虑替代 CNN 中传统神经元模型的人。例如，Jarrett 等人 [11] 声称非线性函数 $f(x) = |tanh(x)|$ 与其对比度归一化一起，然后是局部均值池化，在 Caltech-101 数据集上工作的非常好。然而，在这个数据集上主要的关注点是防止过拟合，因此他们观测到的影响不同于我们使用 ReLU 拟合数据集时的加速能力。更快的学习对大型数据集上大型模型的性能有很大的影响。

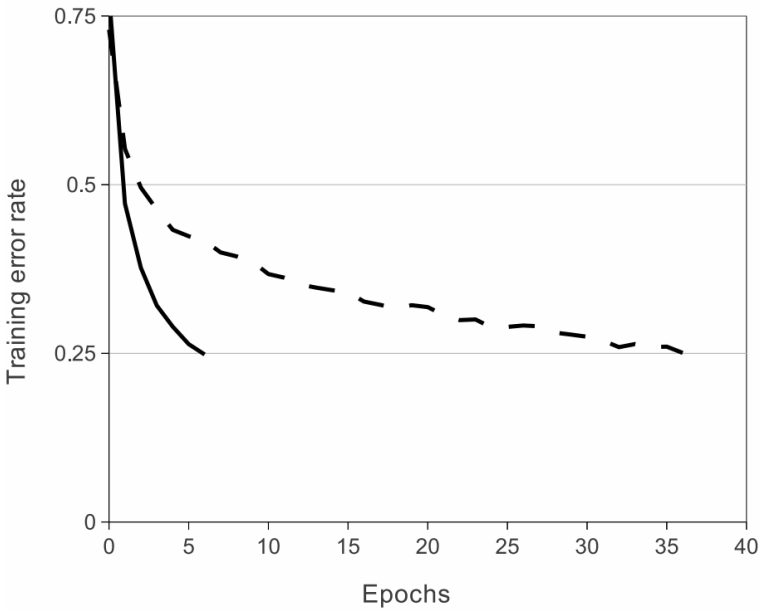


Figure 1: A four-layer convolutional neural network with ReLUs **(solid line)** reaches a 25% training error rate on CIFAR-10 six times faster than an equivalent network with tanh neurons **(dashed line)**. The learning rates for each network were chosen independently to make training as fast as possible. No regularization of any kind was employed. The magnitude of the effect demonstrated here varies with network architecture, but networks with ReLUs consistently learn several times faster than equivalents with saturating neurons.

图 1：使用 ReLU 的四层卷积神经网络在 CIFAR-10 数据集上达到 25% 的训练误差比使用 tanh 神经元的等价网络（虚线）快六倍。为了使训练尽可能快，每个网络的学习率是单独选择的。没有采用任何类型的正则化。影响的大小随着网络结构的变化而变化，这一点已得到证实，但使用 ReLU 的网络都比等价的饱和神经元快几倍。

0.4.2 Training on Multiple GPUs（多 GPU 训练）

A single GTX 580 GPU has only 3GB of memory, which limits the maximum size of the networks that can be trained on it. It turns out that 1.2 million training examples are enough to train networks which are too big to fit on one GPU. Therefore we spread the net across two GPUs. Current GPUs are particularly well-suited to cross-GPU parallelization, as they are able to read from and write to one another's memory directly, without going through host machine memory. The parallelization scheme that we employ essentially puts half of the kernels (or neurons) on each GPU, with one additional trick: the GPUs communicate only in certain layers. This means that, for example, the kernels of layer 3 take input from all kernel maps in layer 2. However, kernels in layer 4 take input only from those kernel maps in layer 3 which reside on the same GPU. Choosing the pattern of connectivity is a problem for cross-validation, but this allows us to precisely tune the amount of communication until it is an acceptable fraction of the amount of computation.

单个 GTX580 GPU 只有 3G 内存，这限制了可以在 GTX580 上进行训练的网络最大尺寸。事实证明 120 万图像用来进行网络训练是足够的，但网络太大因此不能在单个 GPU 上进行训练。因此我们将网络分布在两个 GPU 上。目前的 GPU 非常适合跨 GPU 并行，因为它们可以直接互相读写内存，而不需要通过主机内存。我们采用的并行方案基本上每个 GPU 放置一半的核（或神经元），还有一个额外的技巧：只在某些特定的层上进行 GPU 通信。这意味着，例如，第 3 层的核会将第 2 层的所有核映射作为输入。然而，第 4 层的核只将位于相同 GPU 上的第 3 层的核映射作为输入。连接模式的选择是一个交叉验证问题，但这可以让我们准确地调整通信数量，直到它的计算量在可接受的范围内。

The resultant architecture is somewhat similar to that of the "columnar" CNN employed by Cire¸san et al. [5], except that our columns are not independent (see Figure 2). This scheme reduces our top-1 and top-5 error rates by 1.7% and 1.2%, respectively, as compared with a net with half as many kernels in each convolutional layer trained on one GPU. The two-GPU net takes slightly less time to train than the one-GPU net^2.

除了我们的列不是独立的之外（看图 2），最终的架构有点类似于 Ciresan 等人 [5] 采用的 "columnar" CNN。与每个卷积层一半的核在单 GPU 上训练的网络相比，这个方案降分别低了我们的 top-1 1.7%，top-5 1.2% 的错误率。双 GPU 网络比单 GPU 网络稍微减少了训练时间。

0.4.3 Local Response Normalization（局部响应归一化）

ReLUs have the desirable property that they do not require input normalization to prevent them from saturating. If at least some training examples produce a positive input to a

ReLU, learning will happen in that neuron. However, we still find that the following local normalization scheme aids generalization. Denoting by $a^i_{x,y}$ the activity of a neuron computed by applying kernel i at position (x, y) and then applying the ReLU nonlinearity, the response-normalized activity $b^i_{x,y}$ is given by the expression

ReLU 具有让人满意的特性，它不需要通过输入归一化来防止饱和。如果至少一些训练样本对 ReLU 产生了正输入，那么那个神经元上将发生学习。然而，我们仍然发现接下来的局部响应归一化有助于泛化。$a^i_{x,y}$ 表示神经元激活，通过在 (x, y) 位置应用核 i，然后应用 ReLU 非线性来计算，响应归一化激活 $b^i_{x,y}$ 通过下式给定：

$$b^i_{x,y} = a^i_{x,y} / \left(k + \alpha \sum_{j=max(0,i-n/2)}^{min(N-1,i+n/2)} (a^j_{x,y})^2 \right)^\beta$$

where the sum runs over n "adjacent" kernel maps at the same spatial position, and N is the total number of kernels in the layer. The ordering of the kernel maps is of course arbitrary and determined before training begins. This sort of response normalization implements a form of lateral inhibition inspired by the type found in real neurons, creating competition for big activities amongst neuron outputs computed using different kernels. The constants k, n, α, and β are hyper-parameters whose values are determined using a validation set; we used $k = 2$, $n = 5$, $\alpha = 104$, and $\beta = 0.75$. We applied this normalization after applying the ReLU nonlinearity in certain layers (see Section 3.5).

求和运算在 n 个 "毗邻的" 核映射的同一位置上执行，N 是本层的卷积核数目。核映射的顺序当然是任意的，在训练开始前确定。响应归一化的顺序实现了一种侧抑制形式，灵感来自于真实神经元中发现的类型，为使用不同核进行神经元输出计算的较大活动创造了竞争。常量 k，n，α，β 是超参数，它们的值通过验证集确定；我们设 $k = 2, n = 5, \alpha = 104, \beta = 0.75$。我们在特定的层使用的 ReLU 非线性之后应用了这种归一化（请看 3.5 小节）。

This scheme bears some resemblance to the local contrast normalization scheme of Jarrett et al. [11], but ours would be more correctly termed "brightness normalization", since we do not subtract the mean activity. Response normalization reduces our top-1 and top-5 error rates by 1.4% and 1.2%, respectively. We also verified the effectiveness of this scheme on the CIFAR-10 dataset: a four-layer CNN achieved a 13% test error rate without normalization and 11% with $normalization^3$.

这个方案与 Jarrett 等人 [11] 的局部对比度归一化方案有一定的相似性，但我们更恰当的称其为 "亮度归一化"，因此我们没有减去均值。响应归一化分别减少了 top-1 1.4%，top-5 1.2% 的错误率。我们也在 CIFAR-10 数据集上验证了这个方案的有效性：四层 CNN 在没有标准化的情况下达到 13% 的测试错误率，而在正则的情况下达到 11%。

0.4.4 Overlapping Pooling（重叠池化）

Pooling layers in CNNs summarize the outputs of neighboring groups of neurons in the same kernel map. Traditionally, the neighborhoods summarized by adjacent pooling units do not overlap (e.g., [17, 11, 4]). To be more precise, a pooling layer can be thought of as consisting of a grid of pooling units spaced s pixels apart, each summarizing a neighborhood of size $z \times z$ centered at the location of the pooling unit. If we set $s = z$, we obtain traditional local pooling as commonly employed in CNNs. If we set $s < z$, we obtain overlapping pooling. This is what we use throughout our network, with $s = 2$ and $z = 3$. This scheme reduces the top-1 and top-5 error rates by 0.4% and 0.3%, respectively, as compared with the non-overlapping scheme $s = 2, z = 2$, which produces output of equivalent dimensions. We generally observe during training that models with overlapping pooling find it slightly more difficult to overfit.

CNN 中的池化层归纳了同一核映射上相邻组神经元的输出。习惯上，相邻池化单元归纳的区域是不重叠的（例如 [17, 11, 4]）。更确切的说，池化层可看作由池化单元网格组成，网格间距为 s 个像素，每个网格归纳池化单元中心位置 $z \times z$ 大小的邻居。如果设置 $s = z$，我们会得到通常在 CNN 中采用的传统局部池化。如果设置 $s < z$，我们会得到重叠池化。这就是我们网络中使用的方法，设置 $s = 2$，$z = 3$。这个方案分别降低了 top-1 0.4%，top-5 0.3% 的错误率，与非重叠方案 $s = 2, z = 2$ 相比，输出的维度是相等的。我们在训练过程中通常观察采用重叠池化的模型，发现它更难过拟合。

0.4.5 Overall Architecture（整体架构）

Now we are ready to describe the overall architecture of our CNN. As depicted in Figure 2, the net contains eight layers with weights; the first five are convolutional and the remaining three are fully-connected. The output of the last fully-connected layer is fed to a 1000-way softmax which produces a distribution over the 1000 class labels. Our network maximizes the multinomial logistic regression objective, which is equivalent to maximizing the average across training cases of the log-probability of the correct label under the prediction distribution.

现在我们准备描述我们的 CNN 的整体架构。如图 2 所示，我们的网络包含 8 个带权重的层；前 5 层是卷积层，剩下的 3 层是全连接层。最后一层全连接层的输出是 1000 维 softmax 的输入，softmax 会产生 1000 类标签的分布。我们的网络最大化多项逻辑回归的目标，这等价于最大化预测分布下训练样本正确标签的对数概率的均值。

The kernels of the second, fourth, and fifth convolutional layers are connected only to those kernel maps in the previous layer which reside on the same GPU (see Figure 2). The kernels of the third convolutional layer are connected to all kernel maps in the second layer. The neurons in the fully connected layers are connected to all neurons in the previous layer.

Response-normalization layers follow the first and second convolutional layers. Max-pooling layers, of the kind described in Section 3.4, follow both response-normalization layers as well as the fifth convolutional layer. The ReLU non-linearity is applied to the output of every convolutional and fully-connected layer.

第 2，4，5 卷积层的核只与位于同一 GPU 上的前一层的核映射相连接（看图 2）。第 3 卷积层的核与第 2 层的所有核映射相连。全连接层的神经元与前一层的所有神经元相连。第 1，2 卷积层之后是响应归一化层。3.4 节描述的这种最大池化层在响应归一化层和第 5 卷积层之后。ReLU 非线性应用在每个卷积层和全连接层的输出上。

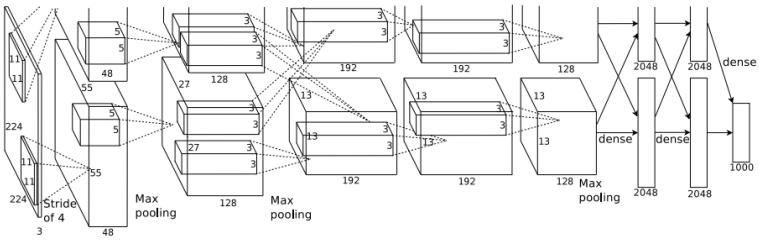


Figure 2: An illustration of the architecture of our CNN, explicitly showing the delineation of responsibilities between the two GPUs. One GPU runs the layer-parts at the top of the figure while the other runs the layer-parts at the bottom. The GPUs communicate only at certain layers. The network's input is 150,528-dimensional, and the number of neurons in the network's remaining layers is given by 253,440–186,624–64,896–64,896–43,264–4096–4096–1000.

The first convolutional layer filters the 224×224×3 input image with 96 kernels of size 11×11×3 with a stride of 4 pixels (this is the distance between the receptive field centers of neighboring neurons in a kernel map). The second convolutional layer takes as input the (response-normalized and pooled) output of the first convolutional layer and filters it with 256 kernels of size 5 × 5 × 48. The third, fourth, and fifth convolutional layers are connected to one another without any intervening pooling or normalization layers. The third convolutional layer has 384 kernels of size 3 × 3 × 256 connected to the (normalized, pooled) outputs of the second convolutional layer. The fourth convolutional layer has 384 kernels of size 3 × 3 × 192 , and the fifth convolutional layer has 256 kernels of size 3 × 3 × 192. The fully-connected layers have 4096 neurons each.

第 1 卷积层使用 96 个核对 224 × 224 × 3 的输入图像进行滤波，核大小为 11 × 11 × 3，步长是 4 个像素（核映射中相邻神经元感受野中心之间的距离）。第 2 卷积层使用用第 1 卷积层的输出（响应归一化和池化）作为输入，并使用 256 个核进行滤波，核大小为 5 × 5 × 48。第 3，4，5 卷积层互相连接，中间没有接入池化层或归一化层。第 3 卷积层有 384 个核，核大小为 3 × 3 × 256，与第 2 卷积层的输出（归一化的，池化的）

相连。第 4 卷积层有 384 个核，核大小为 3 × 3 × 192，第 5 卷积层有 256 个核，核大小为 3 × 3 × 192。每个全连接层有 4096 个神经元。

0.5 Reducing Overfitting（减小过拟合）

Our neural network architecture has 60 million parameters. Although the 1000 classes of ILSVRC make each training example impose 10 bits of constraint on the mapping from image to label, this turns out to be insufficient to learn so many parameters without considerable overfitting. Below, we describe the two primary ways in which we combat overfitting.

我们的神经网络架构有 6000 万参数。尽管 ILSVRC 的 1000 类使每个训练样本从图像到标签的映射上强加了 10 比特的约束，但这不足以学习这么多的参数而没有相当大的过拟合。下面，我们会描述我们用来克服过拟合的两种主要方式。

0.5.1 Data Augmentation（数据增强）

The easiest and most common method to reduce overfitting on image data is to artificially enlarge the dataset using label-preserving transformations (e.g., [25, 4, 5]). We employ two distinct forms of data augmentation, both of which allow transformed images to be produced from the original images with very little computation, so the transformed images do not need to be stored on disk. In our implementation, the transformed images are generated in Python code on the CPU while the GPU is training on the previous batch of images. So these data augmentation schemes are, in effect, computationally free.

图像数据上最简单常用的用来减少过拟合的方法是使用标签保留变换（例如 [25, 4, 5]）来人工增大数据集。我们使用了两种独特的数据增强方式，这两种方式都可以从原始图像通过非常少的计算量产生变换的图像，因此变换图像不需要存储在硬盘上。在我们的实现中，变换图像通过 CPU 的 Python 代码生成，而此时 GPU 正在训练前一批图像。因此，实际上这些数据增强方案是计算免费的。

The first form of data augmentation consists of generating image translations and horizontal reflections. We do this by extracting random 224 × 224 patches (and their horizontal reflections) from the 256×256 images and training our network on these extracted $patches^4$. This increases the size of our training set by a factor of 2048, though the resulting training examples are, of course, highly inter dependent. Without this scheme, our network suffers from substantial overfitting, which would have forced us to use much smaller networks. At test time, the network makes a prediction by extracting five 224 × 224 patches (the four corner patches and the center patch) as well as their horizontal reflections (hence ten patches in all), and averaging the predictions made by the network's softmax layer on the ten patches.

第一种数据增强方式包括产生图像变换和水平翻转。我们从 256×256 图像上通过随机提取 224 × 224 的图像块实现了这种方式，然后在这些提取的图像块上进行训练。

这通过一个 2048 因子增大了我们的训练集，尽管最终的训练样本是高度相关的。没有这个方案，我们的网络会有大量的过拟合，这会迫使我们使用更小的网络。在测试时，网络会提取 5 个 224 × 224 的图像块（四个角上的图像块和中心的图像块）和它们的水平翻转（因此总共 10 个图像块）进行预测，然后对网络在 10 个图像块上的 softmax 层进行平均。

The second form of data augmentation consists of altering the intensities of the RGB channels in training images. Specifically, we perform PCA on the set of RGB pixel values throughout the ImageNet training set. To each training image, we add multiples of the found principal components, with magnitudes proportional to the corresponding eigenvalues times a random variable drawn from a Gaussian with mean zero and standard deviation 0.1. Therefore to each RGB image pixel $I_{xy} = \begin{bmatrix} I_{xy}^R & I_{xy}^G & I_{xy}^B \end{bmatrix}^T$ we add the following quantity:

第二种数据增强方式包括改变训练图像的 RGB 通道的强度。具体地，我们在整个 ImageNet 训练集上对 RGB 像素值集合执行 PCA。对于每幅训练图像，我们加上多倍找到的主成分，大小成正比的对应特征值乘以一个随机变量，随机变量通过均值为 0，标准差为 0.1 的高斯分布得到。因此对于每幅 RGB 图像像素 $I_{xy} = \begin{bmatrix} I_{xy}^R & I_{xy}^G & I_{xy}^B \end{bmatrix}^T$，我们加上下面的数量：

$$\begin{bmatrix} P_1 & P_2 & P_3 \end{bmatrix} \begin{bmatrix} \alpha_1\lambda_1 & \alpha_2\lambda_2 & \alpha_3\lambda_3 \end{bmatrix}^T$$

where P_i and λ_i are ith eigenvector and eigenvalue of the 3 × 3 covariance matrix of RGB pixel values, respectively, and α_i is the aforementioned random variable. Each α_i is drawn only once for all the pixels of a particular training image until that image is used for training again, at which point it is re-drawn. This scheme approximately captures an important property of natural images, namely, that object identity is invariant to changes in the intensity and color of the illumination. This scheme reduces the top-1 error rate by over 1%.

P_i λ_i 分别是 RGB 像素值 3 × 3 协方差矩阵的第 i 个特征向量和特征值，α_i 是前面提到的随机变量。对于某个训练图像的所有像素，每个 α_i 只获取一次，直到图像进行下一次训练时才重新获取。这个方案近似抓住了自然图像的一个重要特性，即光照的颜色和强度发生变化时，目标身份是不变的。这个方案减少了 top 1 错误率 1% 以上。

0.5.2 Dropout

Combining the predictions of many different models is a very successful way to reduce test errors [1, 3], but it appears to be too expensive for big neural networks that already take several days to train. There is, however, a very efficient version of model combination that only costs about a factor of two during training. The recently-introduced technique, called "dropout" [10], consists of setting to zero the output of each hidden neuron with probability

0.5. The neurons which are "dropped out" in this way do not contribute to the forward pass and do not participate in back-propagation. So every time an input is presented, the neural network samples a different architecture, but all these architectures share weights. This technique reduces complex co-adaptations of neurons, since a neuron cannot rely on the presence of particular other neurons. It is, therefore, forced to learn more robust features that are useful in conjunction with many different random subsets of the other neurons. At test time, we use all the neurons but multiply their outputs by 0.5, which is a reasonable approximation to taking the geometric mean of the predictive distributions produced by the exponentially-many dropout networks.

将许多不同模型的预测结合起来是降低测试误差 [1, 3] 的一个非常成功的方法，但对于需要花费几天来训练的大型神经网络来说，这似乎太昂贵了。然而，有一个非常有效的模型结合版本，它只花费两倍的训练成本。这种最近引入的技术，叫做 "dropout"[10]，它会以 0.5 的概率对每个隐层神经元的输出设为 0。那些 "失活的" 的神经元不再进行前向传播并且不参与反向传播。因此每次输入时，神经网络会采样一个不同的架构，但所有架构共享权重。这个技术减少了复杂的神经元互适应，因为一个神经元不能依赖特定的其它神经元的存在。因此，神经元被强迫学习更鲁棒的特征，它在与许多不同的其它神经元的随机子集结合时是有用的。在测试时，我们使用所有的神经元但它们的输出乘以 0.5，对指数级的许多失活网络的预测分布进行几何平均，这是一种合理的近似。

We use dropout in the first two fully-connected layers of Figure 2. Without dropout, our network exhibits substantial overfitting. Dropout roughly doubles the number of iterations required to converge.

我们在图 2 中的前两个全连接层使用失活。如果没有失活，我们的网络表现出大量的过拟合。失活大致上使要求收敛的迭代次数翻了一倍。

0.6 Details of learning（学习细节）

We trained our models using stochastic gradient descent with a batch size of 128 examples, momentum of 0.9, and weight decay of 0.0005. We found that this small amount of weight decay was important for the model to learn. In other words, weight decay here is not merely a regularizer: it reduces the model's training error. The update rule for weight w was

我们使用随机梯度下降来训练我们的模型，样本的 batch size 为 128，动量为 0.9，权重衰减为 0.0005。我们发现少量的权重衰减对于模型的学习是重要的。换句话说，权重衰减不仅仅是一个正则项：它减少了模型的训练误差。权重 w 的更新规则是

$$v_{i+1} := 0.9 \cdot v_i - 0.0005 \cdot \epsilon \cdot w_i - \epsilon \cdot \left\langle \frac{\partial L}{\partial w} |_{w_i} \right\rangle_{D_i}$$

$$w_{i+1} := w_i + v_{i+1}$$

where i is the iteration index, v is the momentum variable, ϵ is the learning rate, and $\left\langle \frac{\partial L}{\partial w} \mid_{w_i} \right\rangle_{D_i}$ is the average over the ith batch D_i of the derivative of the objective with respect to w, evaluated at w_i.

i 是迭代索引，v 是动量变量，ϵ 是学习率，$\left\langle \frac{\partial L}{\partial w} \mid_{w_i} \right\rangle_{D_i}$ 是目标函数对 w，在 w_i 上的第 i 批微分 D_i 的平均。

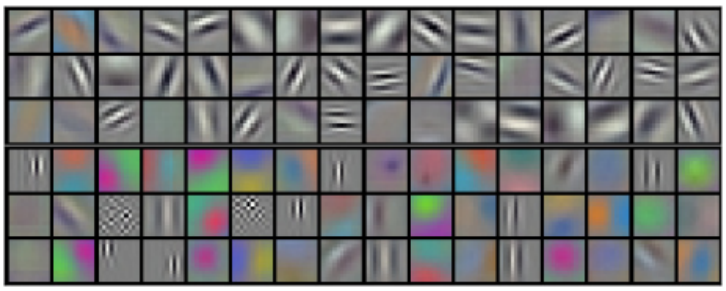


Figure 3: 96 convolutional kernels of size 11×11×3 learned by the first convolutional layer on the 224×224×3 input images. The top 48 kernels were learned on GPU 1 while the bottom 48 kernels were learned on GPU 2. See Section 6.1 for details.

图 3：第一卷积层在 224×224×3 的输入图像上学习到的大小为 11×11×3 的 96 个卷积核。上面的 48 个核是在 GPU 1 上学习到的而下面的 48 个卷积核是在 GPU 2 上学习到的。更多细节请看 6.1 小节。

We initialized the weights in each layer from a zero-mean Gaussian distribution with standard deviation 0.01. We initialized the neuron biases in the second, fourth, and fifth convolutional layers, as well as in the fully-connected hidden layers, with the constant 1. This initialization accelerates the early stages of learning by providing the ReLUs with positive inputs. We initialized the neuron biases in the remaining layers with the constant 0.

我们使用均值为 0，标准差为 0.01 的高斯分布对每一层的权重进行初始化。我们在第 2，4，5 卷积层和全连接隐层将神经元偏置初始化为常量 1。这个初始化通过为 ReLU 提供正输入加速了学习的早期阶段。我们在剩下的层将神经元偏置初始化为 0。

We used an equal learning rate for all layers, which we adjusted manually throughout training. The heuristic which we followed was to divide the learning rate by 10 when the validation error rate stopped improving with the current learning rate. The learning rate was initialized at 0.01 and reduced three times prior to termination. We trained the network for roughly 90 cycles through the training set of 1.2 million images, which took five to six days on two NVIDIA GTX 580 3GB GPUs.

我们对所有的层使用相等的学习率，这个是在整个训练过程中我们手动调整得到的。当验证误差在当前的学习率下停止提供时，我们遵循启发式的方法将学习率除以 10。学习率初始化为 0.01，在训练停止之前降低三次。我们在 120 万图像的训练数据集上训练神经网络大约 90 个循环，在两个 NVIDIA GTX 580 3GB GPU 上花费了五到六天。

0.7 Results（结果）

Our results on ILSVRC-2010 are summarized in Table 1. Our network achieves top-1 and top-5 test set error rates of 37.5% and 17.0. The best performance achieved during the ILSVRC-2010 competition was 47.1% and 28.2% with an approach that averages the predictions produced from six sparse-coding models trained on different features [2], and since then the best published results are 45.7% and 25.7% with an approach that averages the predictions of two classifiers trained on Fisher Vectors (FVs) computed from two types of densely-sampled features [24].

我们在 ILSVRC-2010 上的结果概括为表 1。我们的神经网络取得了 top-1 37.5%，top-5 17.0% 的错误率。在 ILSVRC-2010 竞赛中最佳结果是 top-1 47.1%，top-5 28.2%，使用的方法是对 6 个在不同特征上训练的稀疏编码模型生成的预测进行平均，从那时起已公布的最好结果是 top-1 45.7%，top-5 25.7%，使用的方法是平均在 Fisher 向量（FV）上训练的两个分类器的预测结果，Fisher 向量是通过两种密集采样特征计算得到的 [24]。

Model	**Top-1**	**Top-5**
Sparse coding [2]	47.1%	28.2%
SIFT + FVs [24]	45.7%	25.7%
CNN	**37.5**	**17.0%**

Table 1: Comparison of results on ILSVRC-2010 test set. In italics are best results achieved by others.

表 1：ILSVRC-2010 测试集上的结果对比。斜体是其它人取得的最好结果。

We also entered our model in the ILSVRC-2012 competition and report our results in Table 2. Since the ILSVRC-2012 test set labels are not publicly available, we cannot report test error rates for all the models that we tried. In the remainder of this paragraph, we use validation and test error rates interchangeably because in our experience they do not differ by more than 0.1% (see Table 2). The CNN described in this paper achieves a top-5 error rate of 18.2%. Averaging the predictions of five similar CNNs gives an error rate of 16.4%. Training one CNN, with an extra sixth convolutional layer over the last pooling layer, to classify the

entire ImageNet Fall 2011 release (15M images, 22K categories), and then “fine-tuning” it on ILSVRC-2012 gives an error rate of 16.6%. Averaging the predictions of two CNNs that were pre-trained on the entire Fall 2011 release with the aforementioned five CNNs gives an error rate of 15.3%. The second-best contest entry achieved an error rate of 26.2% with an approach that averages the predictions of several classifiers trained on FVs computed from different types of densely-sampled features [7].

我们也用我们的模型参加了 ILSVRC-2012 竞赛并在表 2 中报告了我们的结果。由于 ILSVRC-2012 的测试集标签不可以公开得到，我们不能报告我们尝试的所有模型的测试错误率。在这段的其余部分，我们会使用验证误差率和测试误差率互换，因为在我们的实验中它们的差别不会超过 0.1%（看图 2）。本文中描述的 CNN 取得了 top-5 18.2% 的错误率。五个类似的 CNN 预测的平均误差率为 16.4%。为了对 ImageNet 2011 秋季发布的整个数据集（1500 万图像，22000 个类别）进行分类，我们在最后的池化层之后有一个额外的第 6 卷积层，训练了一个 CNN，然后在它上面进行 “fine-tuning”，在 ILSVRC-2012 取得了 16.6% 的错误率。对在 ImageNet 2011 秋季发布的整个数据集上预训练的两个 CNN 和前面提到的五个 CNN 的预测进行平均得到了 15.3% 的错误率。第二名的最好竞赛输入取得了 26.2% 的错误率，他的方法是对 FV 上训练的一些分类器的预测结果进行平均，FV 在不同类型密集采样特征计算得到的。

Model	**Top-1(val)**	**Top-5(val)**	**Top-5(test)**
SIFT + FVs [7]	-	-	26.2%
1 CNN	40.7%	18.2%	-
5 CNNs	38.1%	16.4%	**16.4%**
1 CNN*	39.0%	16.6%	-
7 CNNs*	36.7%	15.4%	**15.3%**

Table 2: Comparison of error rates on ILSVRC-2012 validation and test sets. In italics are best results achieved by others. Models with an asterisk* were “pre-trained” to classify the entire ImageNet 2011 Fall release. See Section 6 for details.

表 2：ILSVRC-2012 验证集和测试集的误差对比。斜线部分是其它人取得的最好的结果。带星号的是 “预训练的” 对 ImageNet 2011 秋季数据集进行分类的模型。更多细节请看第六节。

Finally, we also report our error rates on the Fall 2009 version of ImageNet with 10,184 categories and 8.9 million images. On this dataset we follow the convention in the literature of using half of the images for training and half for testing. Since there is no established test set, our split necessarily differs from the splits used by previous authors, but this does not affect the results appreciably. Our top-1 and top-5 error rates on this dataset are 67.4% and

40.9%, attained by the net described above but with an additional, sixth convolutional layer over the last pooling layer. The best published results on this dataset are 78.1% and 60.9% [19].

最后，我们也报告了我们在 ImageNet 2009 秋季数据集上的误差率，ImageNet 2009 秋季数据集有 10,184 个类，890 万图像。在这个数据集上我们按照惯例用一半的图像来训练，一半的图像来测试。由于没有建立测试集，我们的数据集分割有必要不同于以前作者的数据集分割，但这对结果没有明显的影响。我们在这个数据集上的的 top-1 和 top-5 错误率是 67.4% 和 40.9%，使用的是上面描述的在最后的池化层之后有一个额外的第 6 卷积层网络。这个数据集上公开可获得的最好结果是 78.1% 和 60.9%[19]。

0.7.1 Qualitative Evaluations（定性评估）

Figure 3 shows the convolutional kernels learned by the network's two data-connected layers. The network has learned a variety of frequency- and orientation-selective kernels, as well as various colored blobs. Notice the specialization exhibited by the two GPUs, a result of the restricted connectivity described in Section 3.5. The kernels on GPU 1 are largely color-agnostic, while the kernels on on GPU 2 are largely color-specific. This kind of specialization occurs during every run and is independent of any particular random weight initialization (modulo a renumbering of the GPUs).

图 3 显示了网络的两个数据连接层学习到的卷积核。网络学习到了大量的频率核、方向选择核，也学到了各种颜色点。注意两个 GPU 表现出的专业化，3.5 小节中描述的受限连接的结果。GPU 1 上的核主要是没有颜色的，而 GPU 2 上的核主要是针对颜色的。这种专业化在每次运行时都会发生，并且是与任何特别的随机权重初始化（以 GPU 的重新编号为模）无关的。

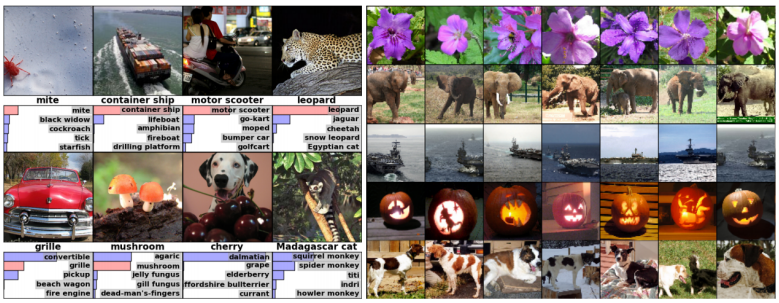


Figure 4: (Left) Eight ILSVRC-2010 test images and the five labels considered most probable by our model. The correct label is written under each image, and the probability assigned to the correct label is also shown with a red bar (if it happens to be in the top 5). (Right) Five ILSVRC-2010 test images in the first column. The remaining columns show the six training images that produce feature vectors in the last hidden layer with the smallest Euclidean distance from the feature vector for the test image.

图 4：（左）8 张 ILSVRC-2010 测试图像和我们的模型认为最可能的 5 个标签。每张图像的下面是它的正确标签，正确标签的概率用红条表示（如果正确标签在 top 5 中）。（右）第一列是 5 张 ILSVRC-2010 测试图像。剩下的列展示了 6 张训练图像，这些图像在最后的隐藏层的特征向量与测试图像的特征向量有最小的欧氏距离。

In the left panel of Figure 4 we qualitatively assess what the network has learned by computing its top-5 predictions on eight test images. Notice that even off-center objects, such as the mite in the top-left, can be recognized by the net. Most of the top-5 labels appear reasonable. For example, only other types of cat are considered plausible labels for the leopard. In some cases (grille, cherry) there is genuine ambiguity about the intended focus of the photograph.

在图 4 的左边部分，我们通过在 8 张测试图像上计算它的 top-5 预测定性地评估了网络学习到的东西。注意即使是不在图像中心的目标也能被网络识别，例如左上角的小虫。大多数的 top-5 标签似乎是合理的。例如，对于美洲豹来说，只有其它类型的猫被认为是看似合理的标签。在某些案例（格栅，樱桃）中，网络在意的图片焦点真的很含糊。

Another way to probe the network's visual knowledge is to consider the feature activations induced by an image at the last, 4096-dimensional hidden layer. If two images produce feature activation vectors with a small Euclidean separation, we can say that the higher levels of the neural network consider them to be similar. Figure 4 shows five images from the test set and the six images from the training set that are most similar to each of them according to this measure. Notice that at the pixel level, the retrieved training images are generally not close in L2 to the query images in the first column. For example, the retrieved dogs and elephants appear in a variety of poses. We present the results for many more test images in the supplementary material.

探索网络可视化知识的另一种方式是思考最后的 4096 维隐藏层在图像上得到的特征激活。如果两幅图像生成的特征激活向量之间有较小的欧式距离，我们可以认为神经网络的更高层特征认为它们是相似的。图 4 表明根据这个度量标准，测试集的 5 张图像和训练集的 6 张图像中的每一张都是最相似的。注意在像素级别，检索到的训练图像与第一列的查询图像在 L2 上通常是不接近的。例如，检索的狗和大象似乎有很多姿态。我们在补充材料中对更多的测试图像呈现了这种结果。

Computing similarity by using Euclidean distance between two 4096-dimensional, real-valued vectors is inefficient, but it could be made efficient by training an auto-encoder to compress these vectors to short binary codes. This should produce a much better image retrieval method than applying auto-encoders to the raw pixels [14], which does not make use of image labels and hence has a tendency to retrieve images with similar patterns of edges, whether or not they are semantically similar

通过两个 4096 维实值向量间的欧氏距离来计算相似性是效率低下的，但通过训练一个自动编码器将这些向量压缩为短二值编码可以使其变得高效。这应该会产生一种比将自动编码器应用到原始像素上 [14] 更好的图像检索方法，自动编码器应用到原始像素上的方法没有使用图像标签，因此会趋向于检索与要检索的图像具有相似边缘模式的图像，无论它们是否是语义上相似。

0.8 Discussion（探讨）

Our results show that a large, deep convolutional neural network is capable of achieving record-breaking results on a highly challenging dataset using purely supervised learning. It is notable that our network's performance degrades if a single convolutional layer is removed. For example, removing any of the middle layers results in a loss of about 2% for the top-1 performance of the network. So the depth really is important for achieving our results.

我们的结果表明一个大型深度卷积神经网络在一个具有高度挑战性的数据集上使用纯有监督学习可以取得破纪录的结果。值得注意的是，如果移除一个卷积层，我们的网络性能会降低。例如，移除任何中间层都会引起网络损失大约 2% 的 top-1 性能。因此深度对于实现我们的结果非常重要。

To simplify our experiments, we did not use any unsupervised pre-training even though we expect that it will help, especially if we obtain enough computational power to significantly increase the size of the network without obtaining a corresponding increase in the amount of labeled data. Thus far, our results have improved as we have made our network larger and trained it longer but we still have many orders of magnitude to go in order to match the infero-temporal pathway of the human visual system. Ultimately we would like to use very large and deep convolutional nets on video sequences where the temporal structure provides very helpful information that is missing or far less obvious in static images.

为了简化我们的实验，我们没有使用任何无监督的预训练，尽管我们希望它会有所帮助，特别是在如果我们能获得足够的计算能力来显著增加网络的大小而标注的数据量没有对应增加的情况下。到目前为止，我们的结果已经提高了，因为我们的网络更大、训练时间更长，但为了匹配人类视觉系统的下颞线（视觉专业术语）我们仍然有许多数量级要达到。最后我们想在视频序列上使用非常大的深度卷积网络，视频序列的时序结构会提供非常有帮助的信息，这些信息在静态图像上是缺失的或远不那么明显。

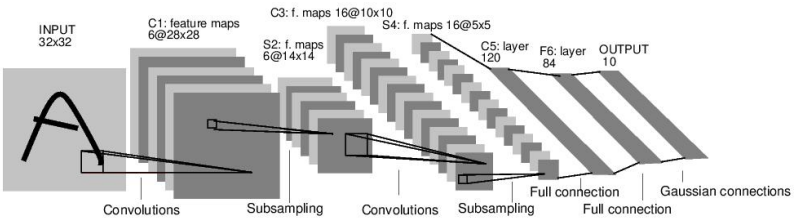
经典深度学习网络模型

0.9 LeNet-5

0.9.1 模型介绍

LeNet-5 是由 $LeCun$ 提出的一种用于识别手写数字和机器印刷字符的卷积神经网络（Convolutional Neural Network，CNN）[1]，其命名来源于作者 $LeCun$ 的名字，5 则是其研究成果的代号，在 LeNet-5 之前还有 LeNet-4 和 LeNet-1 鲜为人知。LeNet-5 阐述了图像中像素特征之间的相关性能够由参数共享的卷积操作所提取，同时使用卷积、下采样（池化）和非线性映射这样的组合结构，是当前流行的大多数深度图像识别网络的基础。

0.9.2 模型结构



LeNet-5 一共包含 7 层（输入层不作为网络结构），分别由 2 个卷积层、2 个下采样层和 3 个全连接层组成，网络的参数配置如下表所示，其中下采样层和全连接层的核尺寸分别代表采样范围和连接矩阵的尺寸（如卷积核尺寸中的 $5 \times 5 \times 1/1, 6$ 表示核大小为 $5 \times 5 \times 1$、步长为 1 且核个数为 6 的卷积核）。

LeNet-5 网络参数配置:

网络层	输入尺寸	核尺寸	输出尺寸	可训练参数量
卷积层 C_1	$32 \times 32 \times 1$	$5 \times 5 \times 1/1, 6$	$28 \times 28 \times 6$	$(5 \times 5 \times 1 + 1) \times 6$
下采样层 S_2	$28 \times 28 \times 6$	$2 \times 2/2$	$14 \times 14 \times 6$	$(1 + 1) \times 6$ *
卷积层 C_3	$14 \times 14 \times 6$	$5 \times 5 \times 6/1, 16$	$10 \times 10 \times 16$	1516*
下采样层 S_4	$10 \times 10 \times 16$	$2 \times 2/2$	$5 \times 5 \times 16$	$(1 + 1) \times 16$
卷积层 C_5*	$5 \times 5 \times 16$	$5 \times 5 \times 16/1, 120$	$1 \times 1 \times 120$	$(5 \times 5 \times 16 + 1) \times 120$
全连接层 F_6	$1 \times 1 \times 120$	120×84	$1 \times 1 \times 84$	$(120 + 1) \times 84$

网络层	输入尺寸	核尺寸	输出尺寸	可训练参数量
输出层	$1 \times 1 \times 84$	84×10	$1 \times 1 \times 10$	$(84+1) \times 10$

* 在 LeNet 中，下采样操作和池化操作类似，但是在得到采样结果后会乘以一个系数和加上一个偏置项，所以下采样的参数个数是 $(1+1) \times 6$ 而不是零。

* C_3 卷积层可训练参数并未直接连接 S_2 中所有的特征图（Feature Map），而是采用如下图所示的采样特征方式进行连接（稀疏连接），生成的 16 个通道特征图中分别按照相邻 3 个特征图、相邻 4 个特征图、非相邻 4 个特征图和全部 6 个特征图进行映射，得到的参数个数计算公式为 $6\times(25\times3+1)+6\times(25\times4+1)+3\times(25\times4+1)+1\times(25\times6+1) = 1516$，在原论文中解释了使用这种采样方式原因包含两点：限制了连接数不至于过大（当年的计算能力比较弱）；强制限定不同特征图的组合可以使映射得到的特征图学习到不同的特征模式。

S_2 与 C_3 之间的特征图稀疏连接

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	X				X	X	X			X	X	X	X		X	X
1	X	X				X	X	X			X	X	X	X		X
2	X	X	X				X	X	X			X		X	X	X
3		X	X	X			X	X	X	X			X		X	X
4			X	X	X			X	X	X	X		X	X		X
5				X	X	X			X	X	X	X		X	X	X

* C_5 卷积层显示为全连接层，原论文中解释这里实际采用的是卷积操作，只是刚好在 5×5 卷积后尺寸被压缩为 1×1，输出结果看起来和全连接很相似。

0.9.3 模型特性

- 卷积网络使用一个 3 层的序列组合：卷积、下采样（池化）、非线性映射（LeNet-5 最重要的特性，奠定了目前深层卷积网络的基础）
- 使用卷积提取空间特征
- 使用映射的空间均值进行下采样
- 使用 $tanh$ 或 $sigmoid$ 进行非线性映射
- 多层神经网络（MLP）作为最终的分类器
- 层间的稀疏连接矩阵以避免巨大的计算开销

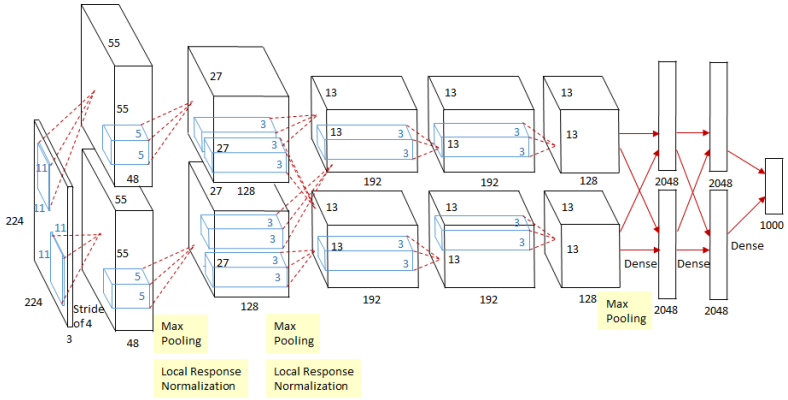
0.10 AlexNet

0.10.1 模型介绍

AlexNet 是由 *Alex Krizhevsky* 提出的首个应用于图像分类的深层卷积神经网络，该网络在 2012 年 ILSVRC（ImageNet Large Scale Visual Recognition Competition）图像

分类竞赛中以 15.3% 的 top-5 测试错误率赢得第一名。AlexNet 使用 GPU 代替 CPU 进行运算，使得在可接受的时间范围内模型结构能够更加复杂，它的出现证明了深层卷积神经网络在复杂模型下的有效性，使 CNN 在计算机视觉中流行开来，直接或间接地引发了深度学习的热潮。

0.10.2 模型结构



除去下采样（池化层）和局部响应规范化操作（Local Responsible Normalization, LRN），AlexNet 一共包含 8 层，前 5 层由卷积层组成，而剩下的 3 层为全连接层。网络结构分为上下两层，分别对应两个 GPU 的操作过程，除了中间某些层（C_3 卷积层和 F_{6-8} 全连接层会有 GPU 间的交互），其他层两个 GPU 分别计算结果。最后一层全连接层的输出作为 $softmax$ 的输入，得到 1000 个图像分类标签对应的概率值。除去 GPU 并行结构的设计，AlexNet 网络结构与 LeNet 十分相似，其网络的参数配置如表所示。

网络层	输入尺寸	核尺寸	输出尺寸	可训练参数量
卷积层 C_1 *	$224 \times 224 \times 3$	$11 \times 11 \times 3/4, 48(\times 2_{GPU})$	$55 \times 55 \times 48(\times 2_{GPU})$	$(11 \times 11 \times 3 + 1) \times 48 \times 2$
下采样层 S_{max}*	$55 \times 55 \times 48(\times 2_{GPU})$	$3 \times 3/2(\times 2_{GPU})$	$27 \times 27 \times 48(\times 2_{GPU})$	0
卷积层 C_2	$27 \times 27 \times 48(\times 2_{GPU})$	$5 \times 5 \times 48/1, 128(\times 2_{GPU})$	$27 \times 27 \times 128(\times 2_{GPU})$	$(5 \times 5 \times 48 + 1) \times 128 \times 2$
下采样层 S_{max}	$27 \times 27 \times 128(\times 2_{GPU})$	$3 \times 3/2(\times 2_{GPU})$	$13 \times 13 \times 128(\times 2_{GPU})$	0
卷积层 C_3 *	$13 \times 13 \times 128 \times 2_{GPU}$	$3 \times 3 \times 256/1, 192(\times 2_{GPU})$	$13 \times 13 \times 192(\times 2_{GPU})$	$(3 \times 3 \times 256 + 1) \times 192 \times 2$
卷积层 C_4	$13 \times 13 \times 192(\times 2_{GPU})$	$3 \times 3 \times 192/1, 192(\times 2_{GPU})$	$13 \times 13 \times 192(\times 2_{GPU})$	$(3 \times 3 \times 192 + 1) \times 192 \times 2$

网络层	输入尺寸	核尺寸	输出尺寸	可训练参数量
卷积层 C_5	$13 \times 13 \times 192(\times 2_{GPU})$	$3 \times 3 \times 192/1,128(\times 2_{GPU})$	$13 \times 13 \times 128(\times 2_{GPU})$	$(3 \times 3 \times 192 + 1) \times 128 \times 2$
下采样层 S_{max}	$13 \times 13 \times 128(\times 2_{GPU})$	$3 \times 3/2(\times 2_{GPU})$	$6 \times 6 \times 128(\times 2_{GPU})$	0
全连接层 F_6 *	$6 \times 6 \times 128 \times 2_{GPU}$	$9216 \times 2048(\times 2_{GPU})$	$1 \times 1 \times 2048(\times 2_{GPU})$	$(9216 + 1) \times 2048 \times 2$
全连接层 F_7	$1 \times 1 \times 2048 \times 2_{GPU}$	$4096 \times 2048(\times 2_{GPU})$	$1 \times 1 \times 2048(\times 2_{GPU})$	$(4096 + 1) \times 2048 \times 2$
全连接层 F_8	$1 \times 1 \times 2048 \times 2_{GPU}$	4096×1000	$1 \times 1 \times 1000$	$(4096 + 1) \times 1000 \times 2$

- 卷积层 C_1 输入为 $224 \times 224 \times 3$ 的图片数据，分别在两个 GPU 中经过核为 $11 \times 11 \times 3$、步长（stride）为 4 的卷积卷积后，分别得到两条独立的 $55 \times 55 \times 48$ 的输出数据。

- 下采样层 S_{max} 实际上是嵌套在卷积中的最大池化操作，但是为了区分没有采用最大池化的卷积层单独列出来。在 C_{1-2} 卷积层中的池化操作之后（ReLU 激活操作之前），还有一个 LRN 操作，用作对相邻特征点的归一化处理。

- 卷积层 C_3 的输入与其他卷积层不同，$13 \times 13 \times 192 \times 2_{GPU}$ 表示汇聚了上一层网络在两个 GPU 上的输出结果作为输入，所以在进行卷积操作时通道上的卷积核维度为 384。

- 全连接层 F_{6-8} 中输入数据尺寸也和 C_3 类似，都是融合了两个 GPU 流向的输出结果作为输入。

0.10.3 模型特性

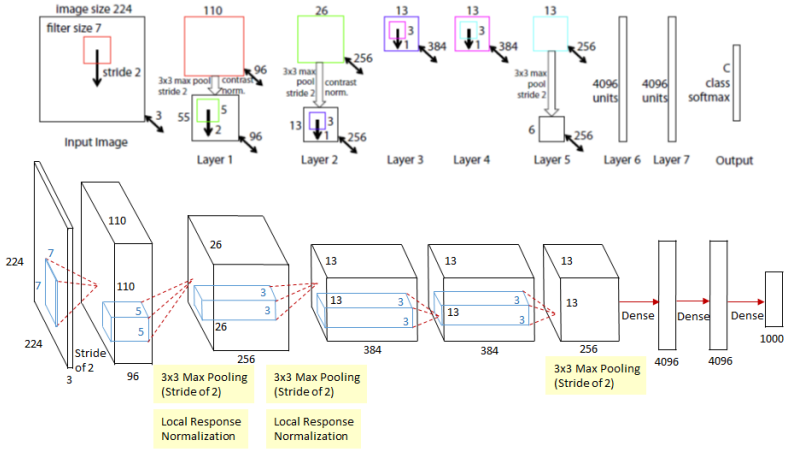
- 所有卷积层都使用 ReLU 作为非线性映射函数，使模型收敛速度更快
- 在多个 GPU 上进行模型的训练，不但可以提高模型的训练速度，还能提升数据的使用规模
- 使用 LRN 对局部的特征进行归一化，结果作为 ReLU 激活函数的输入能有效降低错误率
- 重叠最大池化（overlapping max pooling），即池化范围 z 与步长 s 存在关系 $z > s$（如 S_{max} 中核尺度为 $3 \times 3/2$），避免平均池化（average pooling）的平均效应
- 使用随机丢弃技术（dropout）选择性地忽略训练中的单个神经元，避免模型的过拟合

0.11 ZFNet

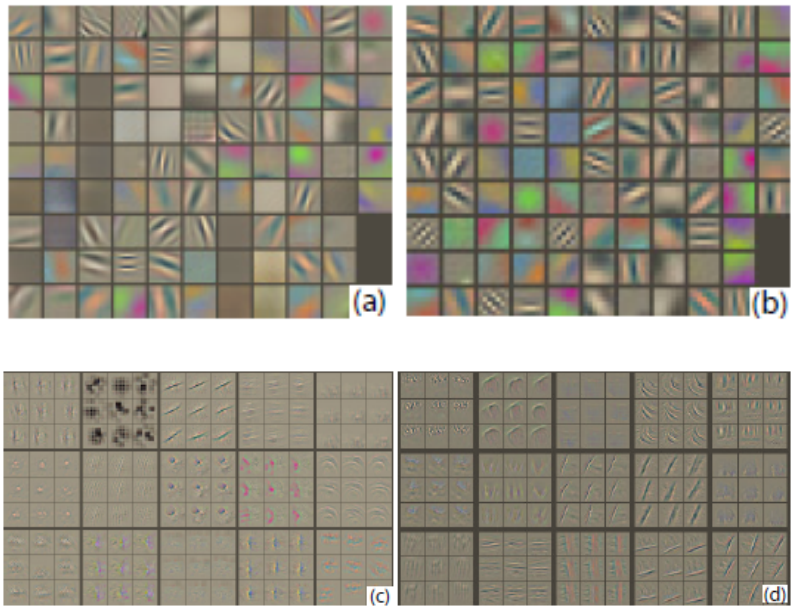
0.11.1 模型介绍

ZFNet 是由 $Matthew\ D.Zeiler$ 和 $Rob\ Fergus$ 在 AlexNet 基础上提出的大型卷积网络，在 2013 年 ILSVRC 图像分类竞赛中以 11.19% 的错误率获得冠军（实际上原 ZFNet 所在的队伍并不是真正的冠军，原 ZFNet 以 13.51% 错误率排在第 8，真正的冠军是 $Clarifai$ 这个队伍，而 $Clarifai$ 这个队伍所对应的一家初创公司的 CEO 又是 $Zeiler$，而且 $Clarifai$ 对 ZFNet 的改动比较小，所以通常认为是 ZFNet 获得了冠军）[3−4]。ZFNet 实际上是微调（fine-tuning）了的 AlexNet，并通过反卷积（Deconvolution）的方式可视化各层的输出特征图，进一步解释了卷积操作在大型网络中效果显著的原因。

0.11.2 模型结构



如图所示，ZFNet 与 AlexNet 类似，都是由 8 层网络组成的卷积神经网络，其中包含 5 层卷积层和 3 层全连接层。两个网络结构最大的不同在于，ZFNet 第一层卷积采用了 $7 \times 7 \times 3/2$ 的卷积核替代了 AlexNet 中第一层卷积核 $11 \times 11 \times 3/4$ 的卷积核。ZFNet 相比于 AlexNet 在第一层输出的特征图中包含更多中间频率的信息，而 AlexNet 第一层输出的特征图大多是低频或高频的信息，对中间频率特征的缺失导致后续网络层次能够学习到的特征不够细致，而导致这个问题的根本原因在于 AlexNet 在第一层中采用的卷积核和步长过大。



（a）ZFNet 第一层输出的特征图（b）AlexNet 第一层输出的特征图（c）AlexNet 第二层输出的特征图（d）ZFNet 第二层输出的特征图

网络层	输入尺寸	核尺寸	输出尺寸	可训练参数量
卷积层 C_1 *	$224 \times 224 \times 3$	$7 \times 7 \times 3/2, 96$	$110 \times 110 \times 96$	$(7 \times 7 \times 3 + 1) \times 96$
下采样层 S_{max}	$110 \times 110 \times 96$	$3 \times 3/2$	$55 \times 55 \times 96$	0
卷积层 C_2 *	$55 \times 55 \times 96$	$5 \times 5 \times 96/2, 256$	$26 \times 26 \times 256$	$(5 \times 5 \times 96 + 1) \times 256$
下采样层 S_{max}	$26 \times 26 \times 256$	$3 \times 3/2$	$13 \times 13 \times 256$	0
卷积层 C_3	$13 \times 13 \times 256$	$3 \times 3 \times 256/1, 384$	$13 \times 13 \times 384$	$(3 \times 3 \times 256 + 1) \times 384$
卷积层 C_4	$13 \times 13 \times 384$	$3 \times 3 \times 384/1, 384$	$13 \times 13 \times 384$	$(3 \times 3 \times 384 + 1) \times 384$
卷积层 C_5	$13 \times 13 \times 384$	$3 \times 3 \times 384/1, 256$	$13 \times 13 \times 256$	$(3 \times 3 \times 384 + 1) \times 256$
下采样层 S_{max}	$13 \times 13 \times 256$	$3 \times 3/2$	$6 \times 6 \times 256$	0
全连接层 F_6	$6 \times 6 \times 256$	9216×4096	$1 \times 1 \times 4096$	$(9216 + 1) \times 4096$

网络层	输入尺寸	核尺寸	输出尺寸	可训练参数量
全连接层 F_7	$1 \times 1 \times 4096$	4096×4096	$1 \times 1 \times 4096$	$(4096+1) \times 4096$
全连接层 F_8	$1 \times 1 \times 4096$	4096×1000	$1 \times 1 \times 1000$	$(4096+1) \times 1000$

- 卷积层 C_1 与 AlexNet 中的 C_1 有所不同，采用 $7 \times 7 \times 3/2$ 的卷积核代替 $11 \times 11 \times 3/4$，使第一层卷积输出的结果可以包含更多的中频率特征，对后续网络层中多样化的特征组合提供更多选择，有利于捕捉更细致的特征。

- 卷积层 C_2 采用了步长 2 的卷积核，区别于 AlexNet 中 C_2 的卷积核步长，所以输出的维度有所差异。

0.11.3 模型特性

ZFNet 与 AlexNet 在结构上几乎相同，此部分虽属于模型特性，但准确地说应该是 ZFNet 原论文中可视化技术的贡献。

- 可视化技术揭露了激发模型中每层单独的特征图。
- 可视化技术允许观察在训练阶段特征的演变过程且诊断出模型的潜在问题。
- 可视化技术用到了多层解卷积网络，即由特征激活返回到输入像素空间。
- 可视化技术进行了分类器输出的敏感性分析，即通过阻止部分输入图像来揭示那部分对于分类是重要的。
- 可视化技术提供了一个非参数的不变性来展示来自训练集的哪一块激活哪个特征图，不仅需要裁剪输入图片，而且自上而下的投影来揭露来自每块的结构激活一个特征图。
- 可视化技术依赖于解卷积操作，即卷积操作的逆过程，将特征映射到像素上。

0.12 Network in Network

0.12.1 模型介绍

Network In Network (NIN) 是由 $MinLin$ 等人提出，在 CIFAR-10 和 CIFAR-100 分类任务中达到当时的最好水平，因其网络结构是由三个多层感知机堆叠而被成为 NIN[5]。NIN 以一种全新的角度审视了卷积神经网络中的卷积核设计，通过引入子网络结构代替纯卷积中的线性映射部分，这种形式的网络结构激发了更复杂的卷积神经网络的结构设计，其中下一节中介绍的 GoogLeNet 的 Inception 结构就是来源于这个思想。

0.12.2 模型结构

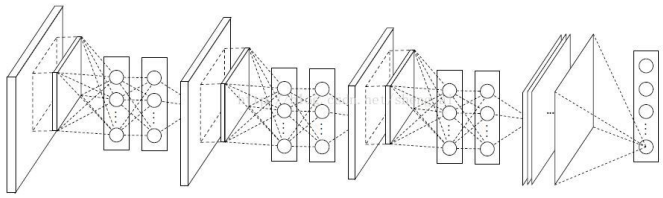


Figure 2: The overall structure of Network In Network. In this paper the NINs include the stacking of three mlpconv layers and one global average pooling layer.

NIN 由三层的多层感知卷积层（MLPConv Layer）构成，每一层多层感知卷积层内部由若干层的局部全连接层和非线性激活函数组成，代替了传统卷积层中采用的线性卷积核。在网络推理（inference）时，这个多层感知器会对输入特征图的局部特征进行划窗计算，并且每个划窗的局部特征图对应的乘积的权重是共享的，这两点是和传统卷积操作完全一致的，最大的不同在于多层感知器对局部特征进行了非线性的映射，而传统卷积的方式是线性的。NIN 的网络参数配置表 4.4 所示（原论文并未给出网络参数，表中参数为编者结合网络结构图和 CIFAR-100 数据集以 3×3 卷积为例给出）。

网络层	输入尺寸	核尺寸	输出尺寸	参数个数
局部全连接层 L_{11} *	$32 \times 32 \times 3$	$(3 \times 3) \times 16/1$	$30 \times 30 \times 16$	$(3 \times 3 \times 3 + 1) \times 16$
全连接层 L_{12} *	$30 \times 30 \times 16$	16×16	$30 \times 30 \times 16$	$((16 + 1) \times 16)$
局部全连接层 L_{21}	$30 \times 30 \times 16$	$(3 \times 3) \times 64/1$	$28 \times 28 \times 64$	$(3 \times 3 \times 16 + 1) \times 64$
全连接层 L_{22}	$28 \times 28 \times 64$	64×64	$28 \times 28 \times 64$	$((64 + 1) \times 64)$
局部全连接层 L_{31}	$28 \times 28 \times 64$	$(3 \times 3) \times 100/1$	$26 \times 26 \times 100$	$(3 \times 3 \times 64 + 1) \times 100$
全连接层 L_{32}	$26 \times 26 \times 100$	100×100	$26 \times 26 \times 100$	$((100 + 1) \times 100)$
全局平均采样 GAP *	$26 \times 26 \times 100$	$26 \times 26 \times 100/1$	$1 \times 1 \times 100$	0

- 局部全连接层 L_{11} 实际上是对原始输入图像进行划窗式的全连接操作，因此划窗得到的输出特征尺寸为 30×30（$\frac{32-3_k+1}{1_{stride}} = 30$）
- 全连接层 L_{12} 是紧跟 L_{11} 后的全连接操作，输入的特征是划窗后经过激活的局部响应特征，因此仅需连接 L_{11} 和 L_{12} 的节点即可，而每个局部全连接层和紧接的

全连接层构成代替卷积操作的多层感知卷积层（MLPConv）。

- 全局平均采样层或全局平均池化层 GAP（Global Average Pooling）将 L_{32} 输出的每一个特征图进行全局的平均池化操作，直接得到最后的类别数，可以有效地减少参数量。

0.12.3 模型特点

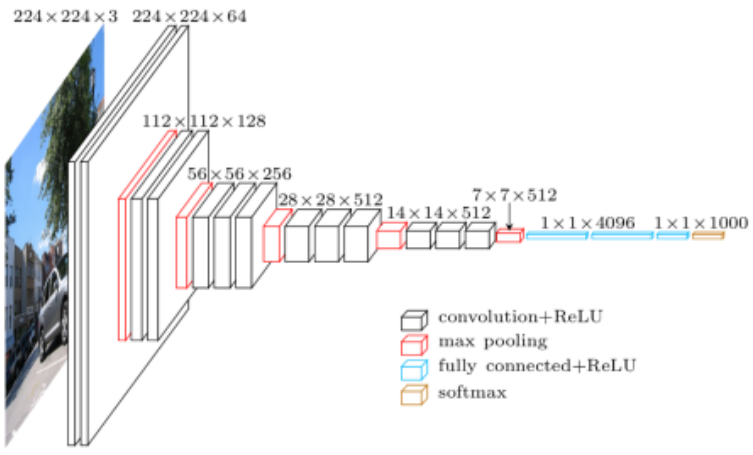
- 使用多层感知机结构来代替卷积的滤波操作，不但有效减少卷积核数过多而导致的参数量暴涨问题，还能通过引入非线性的映射来提高模型对特征的抽象能力。
- 使用全局平均池化来代替最后一个全连接层，能够有效地减少参数量（没有可训练参数），同时池化用到了整个特征图的信息，对空间信息的转换更加鲁棒，最后得到的输出结果可直接作为对应类别的置信度。

0.13 VGGNet

0.13.1 模型介绍

VGGNet 是由牛津大学视觉几何小组（Visual Geometry Group, VGG）提出的一种深层卷积网络结构，他们以 7.32% 的错误率赢得了 2014 年 ILSVRC 分类任务的亚军（冠军由 GoogLeNet 以 6.65% 的错误率夺得）和 25.32% 的错误率夺得定位任务（Localization）的第一名（GoogLeNet 错误率为 26.44%）[5]，网络名称 VGGNet 取自该小组名缩写。VGGNet 是首批把图像分类的错误率降低到 10% 以内模型，同时该网络所采用的 3×3 卷积核的思想是后来许多模型的基础，该模型发表在 2015 年国际学习表征会议（International Conference On Learning Representations, ICLR）后至今被引用的次数已经超过 1 万 4 千余次。

0.13.2 模型结构



在原论文中的 VGGNet 包含了 6 个版本的演进，分别对应 VGG11、VGG11-LRN、

VGG13、VGG16-1、VGG16-3 和 VGG19，不同的后缀数值表示不同的网络层数（VGG11-LRN 表示在第一层中采用了 LRN 的 VGG11，VGG16-1 表示后三组卷积块中最后一层卷积采用卷积核尺寸为 1×1，相应的 VGG16-3 表示卷积核尺寸为 3×3），本节介绍的 VGG16 为 VGG16-3。上图中的 VGG16 体现了 VGGNet 的核心思路，使用 3×3 的卷积组合代替大尺寸的卷积（2 个 3×3 5×5 卷积拥有相同的感受视野），网络参数设置如下表所示。

网络层	输入尺寸	核尺寸	输出尺寸	参数个数
卷积层 C_{11}	$224 \times 224 \times 3$	$3 \times 3 \times 64/1$	$224 \times 224 \times 64$	$(3 \times 3 \times 3 + 1) \times 64$
卷积层 C_{12}	$224 \times 224 \times 64$	$3 \times 3 \times 64/1$	$224 \times 224 \times 64$	$(3 \times 3 \times 64 + 1) \times 64$
下采样层 S_{max1}	$224 \times 224 \times 64$	$2 \times 2/2$	$112 \times 112 \times 64$	0
卷积层 C_{21}	$112 \times 112 \times 64$	$3 \times 3 \times 128/1$	$112 \times 112 \times 128$	$(3 \times 3 \times 64 + 1) \times 128$
卷积层 C_{22}	$112 \times 112 \times 128$	$3 \times 3 \times 128/1$	$112 \times 112 \times 128$	$(3 \times 3 \times 128 + 1) \times 128$
下采样层 S_{max2}	$112 \times 112 \times 128$	$2 \times 2/2$	$56 \times 56 \times 128$	0
卷积层 C_{31}	$56 \times 56 \times 128$	$3 \times 3 \times 256/1$	$56 \times 56 \times 256$	$(3 \times 3 \times 128 + 1) \times 256$
卷积层 C_{32}	$56 \times 56 \times 256$	$3 \times 3 \times 256/1$	$56 \times 56 \times 256$	$(3 \times 3 \times 256 + 1) \times 256$
卷积层 C_{33}	$56 \times 56 \times 256$	$26 \times 26 \times 256/1$	$56 \times 56 \times 256$	$(3 \times 3 \times 256 + 1) \times 256$
下采样层 S_{max3}	$56 \times 56 \times 256$	$2 \times 2/2$	$28 \times 28 \times 256$	0
卷积层 C_{41}	$28 \times 28 \times 256$	$3 \times 3 \times 512/1$	$28 \times 28 \times 512$	$(3 \times 3 \times 256 + 1) \times 512$
卷积层 C_{42}	$28 \times 28 \times 512$	$3 \times 3 \times 512/1$	$28 \times 28 \times 512$	$(3 \times 3 \times 512 + 1) \times 512$
卷积层 C_{43}	$28 \times 28 \times 512$	$3 \times 3 \times 512/1$	$28 \times 28 \times 512$	$(3 \times 3 \times 512 + 1) \times 512$
下采样层 S_{max4}	$28 \times 28 \times 512$	$2 \times 2/2$	$14 \times 14 \times 512$	0

网络层	输入尺寸	核尺寸	输出尺寸	参数个数
卷积层 C_{51}	$14 \times 14 \times 512$	$3 \times 3 \times 512/1$	$14 \times 14 \times 512$	$(3 \times 3 \times 512 + 1) \times 512$
卷积层 C_{52}	$14 \times 14 \times 512$	$3 \times 3 \times 512/1$	$14 \times 14 \times 512$	$(3 \times 3 \times 512 + 1) \times 512$
卷积层 C_{53}	$14 \times 14 \times 512$	$3 \times 3 \times 512/1$	$14 \times 14 \times 512$	$(3 \times 3 \times 512 + 1) \times 512$
下采样层 S_{max5}	$14 \times 14 \times 512$	$2 \times 2/2$	$7 \times 7 \times 512$	0
全连接层 FC_1	$7 \times 7 \times 512$	$(7 \times 7 \times 512) \times 4096$	1×4096	$(7 \times 7 \times 512 + 1) \times 4096$
全连接层 FC_2	1×4096	4096×4096	1×4096	$(4096 + 1) \times 4096$
全连接层 FC_3	1×4096	4096×1000	1×1000	$(4096 + 1) \times 1000$

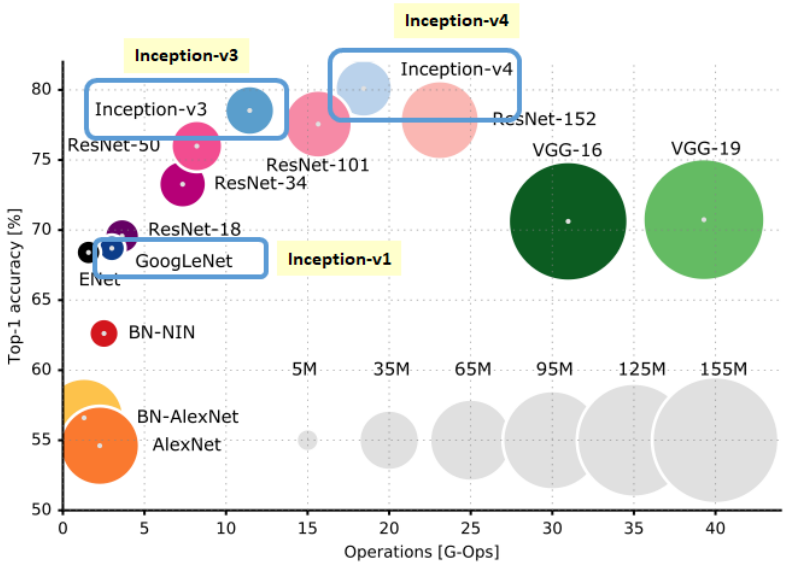
0.13.3 模型特性

- 整个网络都使用了同样大小的卷积核尺寸 3×3 和最大池化尺寸 2×2。
- 1×1 卷积的意义主要在于线性变换，而输入通道数和输出通道数不变，没有发生降维。
- 两个 3×3 的卷积层串联相当于 1 个 5×5 的卷积层，感受野大小为 5×5。同样地，3 个 3×3 的卷积层串联的效果则相当于 1 个 7×7 的卷积层。这样的连接方式使得网络参数量更小，而且多层的激活函数令网络对特征的学习能力更强。
- VGGNet 在训练时有一个小技巧，先训练浅层的的简单网络 VGG11，再复用 VGG11 的权重来初始化 VGG13，如此反复训练并初始化 VGG19，能够使训练时收敛的速度更快。
- 在训练过程中使用多尺度的变换对原始数据做数据增强，使得模型不易过拟合。

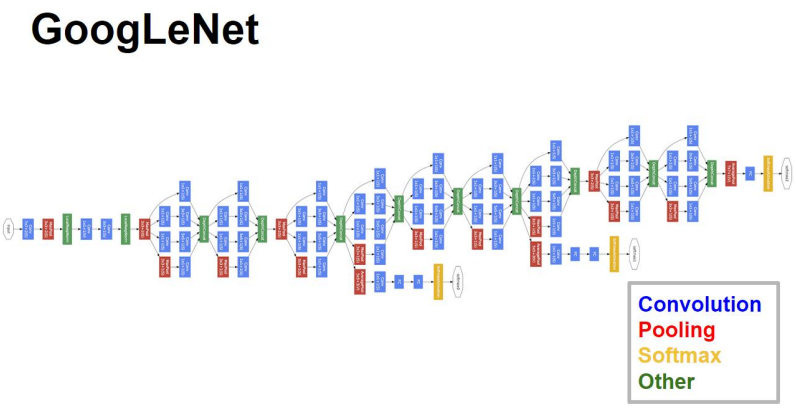
0.14 GoogLeNet

0.14.1 模型介绍

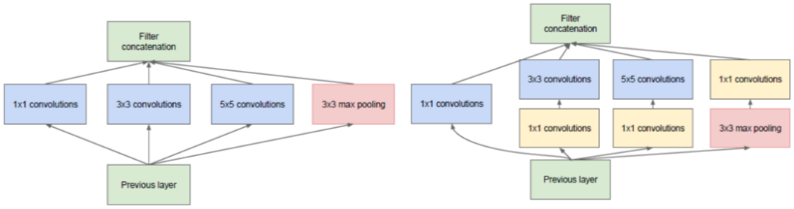
GoogLeNet 作为 2014 年 ILSVRC 在分类任务上的冠军，以 6.65% 的错误率力压 VGGNet 等模型，在分类的准确率上面相比过去两届冠军 ZFNet 和 AlexNet 都有很大的提升。从名字 **GoogLe**Net 可以知道这是来自谷歌工程师所设计的网络结构，而名字中 Goog**LeNet** 更是致敬了 LeNet[0]。GoogLeNet 中最核心的部分是其内部子网络结构 Inception，该结构灵感来源于 NIN，至今已经经历了四次版本迭代（Inception$_{v1-4}$）。

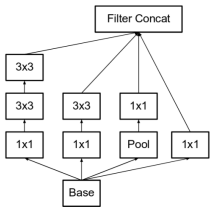


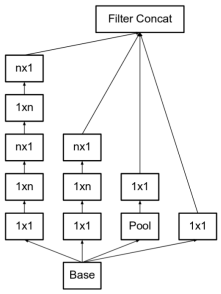
0.14.2 4.6.2 模型结构

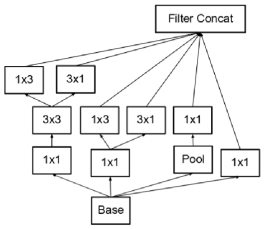


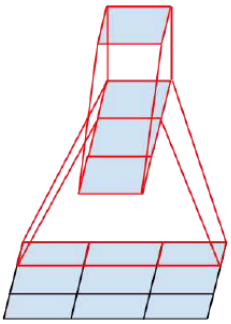
GoogLeNet 相比于以前的卷积神经网络结构，除了在深度上进行了延伸，还对网络的宽度进行了扩展，整个网络由许多块状子网络的堆叠而成，这个子网络构成了 Inception 结构。上图 Inception 的四个版本：$Inception_{v1}$ 在同一层中采用不同的卷积核，并对卷积结果进行合并;$Inception_{v2}$ 组合不同卷积核的堆叠形式，并对卷积结果进行合并;$Inception_{v3}$ 则在 v_2 基础上进行深度组合的尝试;$Inception_{v4}$ 结构相比于前面的版本更加复杂，子网络中嵌套着子网络。

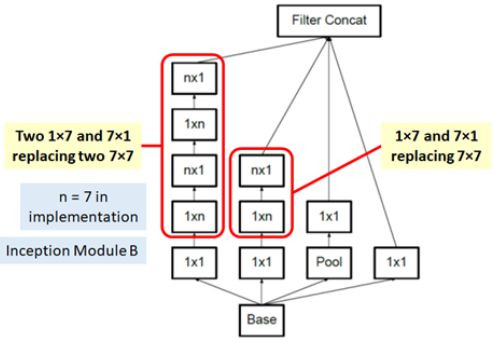


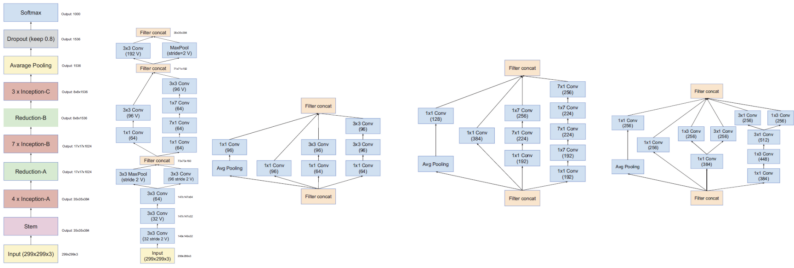












网络层	输入尺寸	核尺寸	输出尺寸	参数个数
卷积层 C_{11}	$H \times W \times C_1$	$1 \times 1 \times C_2/2$	$\frac{H}{2} \times \frac{W}{2} \times C_2$	$(1 \times 1 \times C_1 + 1) \times C_2$
卷积层 C_{21}	$H \times W \times C_2$	$1 \times 1 \times C_2/2$	$\frac{H}{2} \times \frac{W}{2} \times C_2$	$(1 \times 1 \times C_2 + 1) \times C_2$
卷积层 C_{22}	$H \times W \times C_2$	$3 \times 3 \times C_2/1$	$H \times W \times C_2/1$	$(3 \times 3 \times C_2 + 1) \times C_2$
卷积层 C_{31}	$H \times W \times C_1$	$1 \times 1 \times C_2/2$	$\frac{H}{2} \times \frac{W}{2} \times C_2$	$(1 \times 1 \times C_1 + 1) \times C_2$
卷积层 C_{32}	$H \times W \times C_2$	$5 \times 5 \times C_2/1$	$H \times W \times C_2/1$	$(5 \times 5 \times C_2 + 1) \times C_2$
下采样层 S_{41}	$H \times W \times C_1$	$3 \times 3/2$	$\frac{H}{2} \times \frac{W}{2} \times C_2$	0
卷积层 C_{42}	$\frac{H}{2} \times \frac{W}{2} \times C_2$	$1 \times 1 \times C_2/1$	$\frac{H}{2} \times \frac{W}{2} \times C_2$	$(3 \times 3 \times C_2 + 1) \times C_2$

网络层	输入尺寸	核尺寸	输出尺寸	参数个数
合并层 M	$\frac{H}{2} \times \frac{W}{2} \times C_2(\times 4)$	拼接	$\frac{H}{2} \times \frac{W}{2} \times (C_2 \times 4)$	0

0.14.3 模型特性

- 采用不同大小的卷积核意味着不同大小的感受野，最后拼接意味着不同尺度特征的融合；
- 之所以卷积核大小采用 1、3 和 5，主要是为了方便对齐。设定卷积步长 stride=1 之后，只要分别设定 pad=0、1、2，那么卷积之后便可以得到相同维度的特征，然后这些特征就可以直接拼接在一起了；
- 网络越到后面，特征越抽象，而且每个特征所涉及的感受野也更大了，因此随着层数的增加，3x3 和 5x5 卷积的比例也要增加。但是，使用 5x5 的卷积核仍然会带来巨大的计算量。为此，文章借鉴 NIN2，采用 1x1 卷积核来进行降维。

0.15 为什么现在的 CNN 模型都是在 GoogleNet、VGGNet 或者 AlexNet 上调整的？

- 评测对比：为了让自己的结果更有说服力，在发表自己成果的时候会同一个标准的 baseline 及在 baseline 上改进而进行比较，常见的比如各种检测分割的问题都会基于 VGG 或者 Resnet101 这样的基础网络。
- 时间和精力有限：在科研压力和工作压力中，时间和精力只允许大家在有限的范围探索。
- 模型创新难度大：进行基本模型的改进需要大量的实验和尝试，并且需要大量的实验积累和强大灵感，很有可能投入产出比比较小。
- 资源限制：创造一个新的模型需要大量的时间和计算资源，往往在学校和小型商业团队不可行。
- 在实际的应用场景中，其实是有大量的非标准模型的配置。

VERY DEEP CONVOLUTIONAL NETWORKS FOR LARGE-SCALE IMAGE RECOGNITION

0.16 ABSTRACT

In this work we investigate the effect of the convolutional network depth on its accuracy in the large-scale image recognition setting. Our main contribution is a thorough evaluation of networks of increasing depth using an architecture with very small (3×3) convolution filters, which shows that a significant improvement on the prior-art configurations can be achieved by pushing the depth to 16–19 weight layers. These findings were the basis of our ImageNet Challenge 2014 submission, where our team secured the first and the second places in the localisation and classification tracks respectively. We also show that our representations generalise well to other datasets, where they achieve state-of-the-art results. We have made our two best-performing ConvNet models publicly available to facilitate further research on the use of deep visual representations in computer vision.

0.17 INTRODUCTION

Convolutional networks (ConvNets) have recently enjoyed a great success in large-scale image and video recognition (Krizhevsky et al., 2012; Zeiler & Fergus, 2013; Sermanet et al., 2014; Simonyan & Zisserman, 2014) which has become possible due to the large public image repositories, such as ImageNet (Deng et al., 2009), and high-performance computing systems, such as GPUs or large-scale distributed clusters (Dean et al., 2012). In particular, an important role in the advance of deep visual recognition architectures has been played by the ImageNet Large-Scale Visual Recognition Challenge (ILSVRC) (Russakovsky et al., 2014), which has served as a testbed for a few generations of large-scale image classification systems, from high-dimensional shallow feature encodings (Perronnin et al., 2010) (the winner of ILSVRC-2011) to deep ConvNets (Krizhevsky et al., 2012) (the winner of ILSVRC-2012).

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 to facilitate further research.

The rest of the paper is organised as follows. In Sect. 2, we describe our ConvNet configurations. The details of the image classification training and evaluation are then presented in Sect. 3, and the configurations are compared on the ILSVRC classification task in Sect. 4. Sect. 5 concludes the paper. For completeness, we also describe and assess our ILSVRC-2014 object localisation system in Appendix A, and discuss the generalisation of very deep features to other datasets in Appendix B. Finally, Appendix C contains the list of major paper revisions.

0.18 CONVNET CONFIGURATIONS

To measure the improvement brought by the increased ConvNet depth in a fair setting, all our ConvNet layer configurations are designed using the same principles, inspired by Ciresan et al. (2011); Krizhevsky et al. (2012). In this section, we first describe a generic layout of our ConvNet configurations (Sect. 2.1) and then detail the specific configurations used in the evaluation (Sect. 2.2). Our design choices are then discussed and compared to the prior art in Sect. 2.3.

0.18.1 ARCHITECTURE

During training, the input to our ConvNets is a fixed-size 224 × 224 RGB image. The only preprocessing we do is subtracting the mean RGB value, computed on the training set, from each pixel. The image is passed through a stack of convolutional (conv.) layers, where we use filters with a very small receptive field: 3 × 3 (which is the smallest size to capture the notion of left/right, up/down, center). In one of the configurations we also utilise 1 × 1 convolution filters, which can be seen as a linear transformation of the input channels (followed by non-linearity). The convolution stride is fixed to 1 pixel; the spatial padding of conv. layer input is such that the spatial resolution is preserved after convolution, i.e. the padding is 1 pixel for 3 × 3 conv. layers. Spatial pooling is carried out by five max-pooling layers, which follow some of the conv. layers (not all the conv. layers are followed by max-pooling). Max-pooling is performed over a 2 × 2 pixel window, with stride 2.

A stack of convolutional layers (which has a different depth in different architectures)

is followed by three Fully-Connected (FC) layers: the first two have 4096 channels each, the third performs 1000-way ILSVRC classification and thus contains 1000 channels (one for each class). The final layer is the soft-max layer. The configuration of the fully connected layers is the same in all networks.

All hidden layers are equipped with the rectification (ReLU (Krizhevsky et al., 2012)) non-linearity. We note that none of our networks (except for one) contain Local Response Normalisation (LRN) normalisation (Krizhevsky et al., 2012): as will be shown in Sect. 4, such normalisation does not improve the performance on the ILSVRC dataset, but leads to increased memory consumption and computation time. Where applicable, the parameters for the LRN layer are those of (Krizhevsky et al., 2012).

0.18.2 CONFIGURATIONS

The ConvNet configurations, evaluated in this paper, are outlined in Table 1, one per column. In the following we will refer to the nets by their names (A–E). All configurations follow the generic design presented in Sect. 2.1, and differ only in the depth: from 11 weight layers in the network A (8 conv. and 3 FC layers) to 19 weight layers in the network E (16 conv. and 3 FC layers). The width of conv. layers (the number of channels) is rather small, starting from 64 in the first layer and then increasing by a factor of 2 after each max-pooling layer, until it reaches 512.

In Table 2 we report the number of parameters for each configuration. In spite of a large depth, the number of weights in our nets is not greater than the number of weights in a more shallow net with larger conv. layer widths and receptive fields (144M weights in (Sermanet et al., 2014)).

Table 1: **ConvNet configurations** (shown in columns). The depth of the configurations increases from the left (A) to the right (E), as more layers are added (the added layers are shown in bold). The convolutional layer parameters are denoted as "$conv\,\langle receptive field size\rangle - \langle number of channels\rangle$". The ReLU activation function is not shown for brevity.

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					

Table 2: **Number of parameters** (in millions).

Network	A,A-LRN	B	C	D	E
Number of parameters	133	133	134	138	144

0.18.3 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 $3(3^2C^2) = 27C^2$ 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).

The incorporation of 1×1 conv. layers (configuration C, Table 1) is a way to increase the non-linearity of the decision function without affecting the receptive fields of the conv. layers. Even though in our case the 1×1 convolution is essentially a linear projection onto the space of the same dimensionality (the number of input and output channels is the same), an additional non-linearity is introduced by the rectification function. It should be noted that 1×1 conv. layers have recently been utilised in the "Network in Network" architecture of Lin et al. (2014).

Small-size convolution filters have been previously used by Ciresan et al. (2011), but their nets are significantly less deep than ours, and they did not evaluate on the large-scale ILSVRC dataset. Goodfellow et al. (2014) applied deep ConvNets (11 weight layers) to the task of street number recognition, and showed that the increased depth led to better performance. GoogLeNet (Szegedy et al., 2014), a top-performing entry of the ILSVRC-2014 classification task, was developed independently of our work, but is similar in that it is based on very deep ConvNets(22 weight layers) and small convolution filters (apart from 3×3, they also use 1×1 and 5×5 convolutions). Their network topology is, however, more complex than ours, and the spatial resolution of the feature maps is reduced more aggressively in the first layers to decrease the amount of computation. As will be shown in Sect. 4.5, our model is outperforming that of Szegedy et al. (2014) in terms of the single-network classification accuracy.

0.19 CLASSIFICATION FRAMEWORK

In the previous section we presented the details of our network configurations. In this section, we describe the details of classification ConvNet training and evaluation.

0.19.1 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 $5 \mathring{u} 10^4$) 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 102 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).

To obtain the fixed-size 224×224 ConvNet input images, they were randomly cropped from rescaled training images (one crop per image per SGD iteration). To further augment the training set, the crops underwent random horizontal flipping and random RGB colour shift (Krizhevsky et al., 2012). Training image rescaling is explained below.

Training image size. Let S be the smallest side of an isotropically-rescaled training image, from which the ConvNet input is cropped (we also refer to S as the training scale). While the crop size is fixed to 224 × 224, in principle S can take on any value not less than 224: for S = 224 the crop will capture whole-image statistics, completely spanning the smallest side of a training image; for S □ 224 the crop will correspond to a small part of the image, containing a small object or an object part.

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 model is trained to recognise objects over a wide range of scales. For speed reasons, we trained multi-scale models by fine-tuning all layers of a single-scale model with the same configuration, pre-trained with fixed $S = 384$.

0.19.2 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).

0.19.3 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.

0.20 CLASSIFICATION EXPERIMENTS

Dataset. In this section, we present the image classification results achieved by the described ConvNet architectures on the ILSVRC-2012 dataset (which was used for ILSVRC 2012–2014 challenges). The dataset includes images of 1000 classes, and is split into three sets: training (1.3M images), validation (50K images), and testing (100K images with held-out class labels). The classification performance is evaluated using two measures: the top-1 and top-5 error. The former is a multi-class classification error, i.e. the proportion of incorrectly classified images; the latter is the main evaluation criterion used in ILSVRC, and is computed as the proportion of images such that the ground-truth category is outside the top-5 predicted categories.

For the majority of experiments, we used the validation set as the test set. Certain experiments were also carried out on the test set and submitted to the official ILSVRC server as a “VGG” team entry to the ILSVRC-2014 competition (Russakovsky et al., 2014).

0.20.1 SINGLE SCALE EVALUATION

We begin with evaluating the performance of individual ConvNet models at a single scale with the layer configurations described in Sect. 2.2. The test image size was set as follows: $Q = S$ for fixed S, and $Q = 0.5(S_{min} + S_{max})$ for jittered $S \in [S_{min}, S_{max}]$. The results of are shown in Table 3.

First, we note that using local response normalisation (A-LRN network) does not improve on the model A without any normalisation layers. We thus do not employ normalisation in the deeper architectures (B–E).

Second, we observe that the classification error decreases with the increased ConvNet depth: from 11 layers in A to 19 layers in E. Notably, in spite of the same depth, the configuration C (which contains three 1 × 1 conv. layers), performs worse than the configuration D, which uses 3 × 3 conv. layers throughout the network. This indicates that while the additional non-linearity does help (C is better than B), it is also important to capture spatial context by using conv. filters with non-trivial receptive fields (D is better than C). The error rate of our architecture saturates when the depth reaches 19 layers, but even deeper models might be beneficial for larger datasets. We also compared the net B with a shallow net with five 5 × 5 conv. layers, which was derived from B by replacing each pair of 3 × 3 conv. layers with a single 5 × 5 conv. layer (which has the same receptive field as explained in Sect. 2.3). The top-1 error of the shallow net was measured to be 7% higher than that of B (on a center crop), which confirms that a deep net with small filters outperforms a shallow net with larger filters.

Finally, scale jittering at training time ($S \in [256; 512]$) leads to significantly better results than training on images with fixed smallest side ($S = 256$ or $S = 384$), even though a single scale is used at test time. This confirms that training set augmentation by scale jittering is indeed helpful for capturing multi-scale image statistics.

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**

0.20.2 MULTI-SCALE EVALUATION

Having evaluated the ConvNet models at a single scale, we now assess the effect of scale jittering at test time. It consists of running a model over several rescaled versions of a test image (corresponding to different values of Q), followed by averaging the resulting class posteriors. Considering that a large discrepancy between training and testing scales leads to a drop in performance, the models trained with fixed S were evaluated over three test image sizes, close to the training one: $Q = S32, S, S + 32$. At the same time, scale jittering at training time allows the network to be applied to a wider range of scales at test time, so the model trained with variable $S \in [S_{min}; S_{max}]$ was evaluated over a larger range of sizes

$Q = S_{min}, 0.5(S_{min} + S_{max}), S_{max}$.

The results, presented in Table 4, indicate that scale jittering at test time leads to better performance (as compared to evaluating the same model at a single scale, shown in Table 3). As before, the deepest configurations (D and E) perform the best, and scale jittering is better than training with a fixed smallest side S. Our best single-network performance on the validation set is 24.8%/7.5% top-1/top-5 error (highlighted in bold in Table 4). On the test set, the configuration E achieves 7.3% top-5 error.

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**

0.20.3 MULTI-CROP EVALUATION

In Table 5 we compare dense ConvNet evaluation with mult-crop evaluation (see Sect. 3.2 for details). We also assess the complementarity of the two evaluation techniques by averaging their softmax outputs. As can be seen, using multiple crops performs slightly better than dense evaluation, and the two approaches are indeed complementary, as their combination outperforms each of them. As noted above, we hypothesize that this is due to a different treatment of convolution boundary conditions.

Table 5: **ConvNet evaluation techniques comparison.** In all experiments the training scale S was sampled from [256; 512], and three test scales Q were considered: {256, 384, 512}.

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**

0.20.4 CONVNET FUSION

Up until now, we evaluated the performance of individual ConvNet models. In this part of the experiments, we combine the outputs of several models by averaging their soft-max class posteriors. This improves the performance due to complementarity of the models, and was used in the top ILSVRC submissions in 2012 (Krizhevsky et al., 2012) and 2013 (Zeiler & Fergus, 2013; Sermanet et al., 2014).

The results are shown in Table 6. By the time of ILSVRC submission we had only trained the single-scale networks, as well as a multi-scale model D (by fine-tuning only the

fully-connected layers rather than all layers). The resulting ensemble of 7 networks has 7.3% ILSVRC test error. After the submission, we considered an ensemble of only two best-performing multi-scale models (configurations D and E), which reduced the test error to 7.0% using dense evaluation and 6.8% using combined dense and multi-crop evaluation. For reference, our best-performing single model achieves 7.1% error (model E, Table 5).

0.20.5 COMPARISON WITH THE STATE OF THE ART

Finally, we compare our results with the state of the art in Table 7. In the classification task of ILSVRC-2014 challenge (Russakovsky et al., 2014), our "VGG" team secured the 2nd place with 7.3% test error using an ensemble of 7 models. After the submission, we decreased the error rate to 6.8% using an ensemble of 2 models.

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**

As can be seen from Table 7, our very deep ConvNets significantly outperform the previous generation of models, which achieved the best results in the ILSVRC-2012 and ILSVRC-2013 competitions. Our result is also competitive with respect to the classification task winner (GoogLeNet with 6.7% error) and substantially outperforms the ILSVRC-2013 winning submission Clarifai, which achieved 11.2% with outside training data and 11.7% without it. This is remarkable, considering that our best result is achieved by combining just two models – significantly less than used in most ILSVRC submissions. In terms of the single-net performance, our architecture achieves the best result (7.0% test error), outperforming a single GoogLeNet by 0.9%. Notably, we did not depart from the classical ConvNet architecture of LeCun et al. (1989), but improved it by substantially increasing the depth.

Table 7: **Comparison with the state of the art in ILSVRC classification.** Our method is denoted as "VGG". Only the results obtained without outside training data are reported.

Method	top-1 val. error (%)	top-5 val. error (%)	top-5 test error (%)
VGG (2 nets, multi-crop & dense eval.)	**23.7**	**6.8**	**6.8**
VGG (1 net, multi-crop & dense eval.)	24.4	7.1	7.0
VGG (ILSVRC submission, 7 nets, dense eval.)	24.7	7.5	7.3
GoogLeNet (Szegedy et al., 2014) (1 net)	-	7.9	
GoogLeNet (Szegedy et al., 2014) (7 nets)	-	**6.7**	
MSRA (He et al., 2014) (11 nets)	-	-	8.1
MSRA (He et al., 2014) (1 net)	27.9	9.1	9.1
Clarifai (Russakovsky et al., 2014) (multiple nets)	-	-	11.7
Clarifai (Russakovsky et al., 2014) (1 net)	-	-	12.5
Zeiler & Fergus (Zeiler & Fergus, 2013) (6 nets)	36.0	14.7	14.8
Zeiler & Fergus (Zeiler & Fergus, 2013) (1 net)	37.5	16.0	16.1
OverFeat (Sermanet et al., 2014) (7 nets)	34.0	13.2	13.6
OverFeat (Sermanet et al., 2014) (1 net)	35.7	14.2	-
Krizhevsky et al. (Krizhevsky et al., 2012) (5 nets)	38.1	16.4	16.4
Krizhevsky et al. (Krizhevsky et al., 2012) (1 net)	40.7	18.2	-

0.21 CONCLUSION

In this work we evaluated very deep convolutional networks (up to 19 weight layers) for large-scale image classification. It was demonstrated that the representation depth is beneficial for the classification accuracy, and that state-of-the-art performance on the ImageNet challenge dataset can be achieved using a conventional ConvNet architecture (LeCun et al., 1989; Krizhevsky et al., 2012) with substantially increased depth. In the appendix, we also show that our models generalise well to a wide range of tasks and datasets, matching or outperforming more complex recognition pipelines built around less deep image representations. Our results yet again confirm the importance of depth in visual representations.

ACKNOWLEDGEMENTS This work was supported by ERC grant VisRec no. 228180. We gratefully acknowledge the support of NVIDIA Corporation with the donation of the GPUs used for this research.

REFERENCES

Bell, S., Upchurch, P., Snavely, N., and Bala, K. Material recognition in the wild with the materials in context database. CoRR, abs/1412.0623, 2014.

Chatfield, K., Simonyan, K., Vedaldi, A., and Zisserman, A. Return of the devil in the details: Delving deep into convolutional nets. In Proc. BMVC., 2014.

Cimpoi, M., Maji, S., and Vedaldi, A. Deep convolutional filter banks for texture recognition and segmentation. CoRR, abs/1411.6836, 2014.

Ciresan, D. C., Meier, U., Masci, J., Gambardella, L. M., and Schmidhuber, J. Flexible, high performance convolutional neural networks for image classification. In IJCAI, pp. 1237–1242, 2011.

Dean, J., Corrado, G., Monga, R., Chen, K., Devin, M., Mao, M., Ranzato, M., Senior, A., Tucker, P., Yang, K., Le, Q. V., and Ng, A. Y. Large scale distributed deep networks. In NIPS, pp. 1232–1240, 2012.

Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., and Fei-Fei, L. Imagenet: A large-scale hierarchical image database. In Proc. CVPR, 2009.

Donahue, J., Jia, Y., Vinyals, O., Hoffman, J., Zhang, N., Tzeng, E., and Darrell, T. Decaf: A deep convolutional activation feature for generic visual recognition. CoRR, abs/1310.1531, 2013.

Everingham, M., Eslami, S. M. A., Van Gool, L., Williams, C., Winn, J., and Zisserman, A. The Pascal visual object classes challenge: A retrospective. IJCV, 111(1):98–136, 2015.

Fei-Fei, L., Fergus, R., and Perona, P. Learning generative visual models from few training examples: An incremental bayesian approach tested on 101 object categories. In IEEE CVPR Workshop of Generative Model Based Vision, 2004.

Girshick, R. B., Donahue, J., Darrell, T., and Malik, J. Rich feature hierarchies for accu-

rate object detection and semantic segmentation. CoRR, abs/1311.2524v5, 2014. Published in Proc. CVPR, 2014.

Gkioxari, G., Girshick, R., and Malik, J. Actions and attributes from wholes and parts. CoRR, abs/1412.2604, 2014.

Glorot, X. and Bengio, Y. Understanding the difficulty of training deep feedforward neural networks. In Proc. AISTATS, volume 9, pp. 249–256, 2010.

Goodfellow, I. J., Bulatov, Y., Ibarz, J., Arnoud, S., and Shet, V. Multi-digit number recognition from street view imagery using deep convolutional neural networks. In Proc. ICLR, 2014.

Griffin, G., Holub, A., and Perona, P. Caltech-256 object category dataset. Technical Report 7694, California Institute of Technology, 2007.

He, K., Zhang, X., Ren, S., and Sun, J. Spatial pyramid pooling in deep convolutional networks for visual recognition. CoRR, abs/1406.4729v2, 2014.

Hoai, M. Regularized max pooling for image categorization. In Proc. BMVC., 2014.

Howard, A. G. Some improvements on deep convolutional neural network based image classification. In Proc. ICLR, 2014.

Jia, Y. Caffe: An open source convolutional architecture for fast feature embedding. http://caffe.berkeleyvision.org/, 2013.

Karpathy, A. and Fei-Fei, L. Deep visual-semantic alignments for generating image descriptions. CoRR, abs/1412.2306, 2014.

Kiros, R., Salakhutdinov, R., and Zemel, R. S. Unifying visual-semantic embeddings with multimodal neural language models. CoRR, abs/1411.2539, 2014.

Krizhevsky, A. One weird trick for parallelizing convolutional neural networks. CoRR, abs/1404.5997, 2014.

Krizhevsky, A., Sutskever, I., and Hinton, G. E. ImageNet classification with deep convolutional neural networks. In NIPS, pp. 1106–1114, 2012.

LeCun, Y., Boser, B., Denker, J. S., Henderson, D., Howard, R. E., Hubbard, W., and Jackel, L. D. Backpropagation applied to handwritten zip code recognition. Neural Computation, 1(4):541–551, 1989.

Lin, M., Chen, Q., and Yan, S. Network in network. In Proc. ICLR, 2014.

Long, J., Shelhamer, E., and Darrell, T. Fully convolutional networks for semantic segmentation. CoRR, abs/1411.4038, 2014.

Oquab, M., Bottou, L., Laptev, I., and Sivic, J. Learning and Transferring Mid-Level Image Representations using Convolutional Neural Networks. In Proc. CVPR, 2014.

Perronnin, F., Sa´nchez, J., and Mensink, T. Improving the Fisher kernel for large-scale image classification. In Proc. ECCV, 2010.

Razavian, A., Azizpour, H., Sullivan, J., and Carlsson, S. CNN Features off-the-shelf: an Astounding Baseline for Recognition. CoRR, abs/1403.6382, 2014.

Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., Berg, A. C., and Fei-Fei, L. ImageNet large scale visual recognition challenge. CoRR, abs/1409.0575, 2014.

Sermanet, P., Eigen, D., Zhang, X., Mathieu, M., Fergus, R., and LeCun, Y. OverFeat: Integrated Recognition, Localization and Detection using Convolutional Networks. In Proc. ICLR, 2014.

Simonyan, K. and Zisserman, A. Two-stream convolutional networks for action recognition in videos. CoRR, abs/1406.2199, 2014. Published in Proc. NIPS, 2014.

Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., and Rabinovich, A. Going deeper with convolutions. CoRR, abs/1409.4842, 2014.

Wei, Y., Xia, W., Huang, J., Ni, B., Dong, J., Zhao, Y., and Yan, S. CNN: Single-label to multi-label. CoRR, abs/1406.5726, 2014.

Zeiler, M. D. and Fergus, R. Visualizing and understanding convolutional networks. CoRR, abs/1311.2901, 2013. Published in Proc. ECCV, 2014.