ImageNet Classification with Deep Convolutional Neural Networks

Alex Krizhevsky
University of Toronto
kriz@cs.utoronto.ca

Ilya Sutskever
University of Toronto
ilya@cs.utoronto.ca

Geoffrey E. Hinton
University of Toronto
hinton@cs.utoronto.ca

Abstract

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.

1 Introduction

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 current-best error rate on the MNIST digit-recognition task (<0.3%) approaches human performance [4]. But objects in realistic settings exhibit considerable 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.

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.

使用深度卷积神经网络进行ImageNet分类

亚历克斯·克里泽夫斯基 多伦多大学 kriz@cs.utoronto.ca

伊利亚·苏茨克维 多伦多大学 ilya@cs.utoronto.ca

杰弗里·E·辛顿 多伦多大学 hinton@cs.utoronto.ca

摘要

我们训练了一个大型深度卷积神经网络，将ImageNet LSVRC-2010竞赛中的120万张高分辨率图像分类为1000个不同的类别。在测试数据上，我们取得了37.5%和17.0%的top-1和top-5错误率，这比先前的最佳结果有显著提升。该神经网络包含6000万个参数和65万个神经元，由五个卷积层（其中一些后面连接了最大池化层）和三个全连接层（最终通过一个1000维的softmax进行输出）组成。为了加快训练速度，我们使用了非饱和神经元，并对卷积运算采用了高效的GPU实现。为了减少全连接层中的过拟合，我们采用了一种名为“dropout”的正则化方法，该方法被证明非常有效。我们还将该模型的一个变体参加了ILSVRC-2012竞赛，并以15.3%的top-5测试错误率获胜，而第二名的最佳成绩为26.2%。

1 引言

当前的目标识别方法主要依赖于机器学习技术。为了提升其性能，我们可以收集更大规模的数据集、学习更强大的模型，并采用更有效的技术来防止过拟合。直到最近，带标签图像数据集的规模仍相对较小——大约在数万张图像的量级（例如NORB [16]、Caltech-101/256 [8, 9]和CIFAR-10/100 [12]）。对于简单的识别任务，这类规模的数据集已能取得相当好的效果，尤其是在通过标签保持变换进行数据增强的情况下。例如，MNIST手写数字识别任务当前的最佳错误率（<0.3%）已接近人类水平[4]。然而，现实场景中的物体存在显著多样性，因此要学习识别它们必须使用更庞大的训练集。事实上，小规模图像数据集的缺陷早已被广泛认知（如Pinto等人[21]所述），但直到近期才得以收集到包含数百万张图像的标注数据集。新出现的大规模数据集包括LabelMe[23]（包含数十万张完全分割的图像）和ImageNet[6]（涵盖超过22,000个类别、1500多万张带标签的高分辨率图像）。

为了从数百万张图像中学习成千上万的物体，我们需要一个具备强大学习能力的模型。然而，物体识别任务的极端复杂性意味着，即使像ImageNet这样庞大的数据集也无法完全界定这一问题，因此我们的模型还应具备大量先验知识，以弥补我们未掌握的数据。卷积神经网络（CNNs）正是这样一类模型[16, 11, 13, 18, 15, 22, 26]。其能力可通过调整深度和宽度进行控制，并且它们对图像本质（即统计的平稳性和像素依赖的局部性）作出了强有力且基本正确的假设。因此，与具有相似规模层的标准前馈神经网络相比，CNNs的连接和参数数量要少得多，从而更易于训练，而其理论最佳性能可能仅略逊一筹。

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.

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[1]. 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.

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.

2 The Dataset

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.

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.

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.

3 The Architecture

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.

[1]http://code.google.com/p/cuda-convnet/

尽管卷积神经网络（CNN）具有诱人的特性，且其局部架构相对高效，但将其大规模应用于高分辨率图像的成本仍然高得令人望而却步。幸运的是，当前的GPU与高度优化的二维卷积实现相结合，已足够强大，能够支持训练规模可观的CNN；而近期的数据集（如ImageNet）包含了充足的标注样本，足以训练此类模型而不会产生严重的过拟合。

本文的具体贡献如下：我们在ILSVRC-2010和ILSVRC-2012竞赛[2]使用的ImageNet子集上训练了迄今为止最大的卷积神经网络之一，并取得了这些数据集上迄今报道的最佳结果。我们编写了高度优化的二维卷积GPU实现以及训练卷积神经网络所需的所有其他操作，并已公开提供[1]。我们的网络包含多项新颖且非常规的特性，这些特性提升了其性能并缩短了训练时间，详见第3节。即使拥有120万张带标签的训练样本，网络的规模仍使过拟合成为显著问题，因此我们采用了多种有效技术来防止过拟合，详见第4节。最终的网络包含五个卷积层和三个全连接层，这种深度至关重要：我们发现移除任何卷积层（每层参数不超过模型总参数的1%）都会导致性能下降。

最终，网络的规模主要受限于当前GPU的可用内存量以及我们愿意容忍的训练时间。我们的网络在两块GTX 580 3GB GPU上训练需要五到六天。我们所有的实验都表明，只需等待更快的GPU和更大的数据集出现，我们的结果就能得到提升。

2 数据集

ImageNet是一个包含超过1500万张标注高分辨率图像的数据集，涵盖约22,000个类别。这些图像从网络收集，并通过亚马逊众包工具Mechanical Turk由人工标注。自2010年起，作为Pascal视觉对象挑战赛的一部分，每年举办名为ImageNet大规模视觉识别挑战赛（ILSVRC）的竞赛。ILSVRC使用ImageNet的子集，包含1000个类别，每个类别约1000张图像。总计约120万张训练图像、5万张验证图像和15万张测试图像。

ILSVRC-2010是唯一一个测试集标签可公开获取的ILSVRC版本，因此我们的大部分实验均基于此版本进行。由于我们也参加了ILSVRC-2012竞赛，在第6节中我们同样报告了模型在该版本数据集上的结果——该版本的测试集标签未公开。在ImageNet上通常报告两种错误率：top-1和top-5，其中top-5错误率指模型预测概率最高的五个标签中均未包含正确标签的测试图像比例。

ImageNet由可变分辨率的图像组成，而我们的系统需要恒定的输入维度。因此，我们将图像下采样至固定的256×256分辨率。对于矩形图像，我们首先重新缩放图像，使其短边长度为256，然后从结果图像中裁剪出中心的256×256区域。除了从每个像素中减去训练集上的平均活跃度外，我们没有以任何其他方式预处理图像。因此，我们直接在像素的（中心化）原始RGB值上训练了网络。

3 架构

我们的网络架构如图2所示。它包含八个学习层——五个卷积层和三个全连接层。接下来，我们将描述我们网络架构中一些新颖或不寻常的特点。章节3.1-3.4按照我们对其重要性的评估进行排序，最重要的部分排在前面。

[1]http://code.google.com/p/cuda-convnet/

3.1 ReLU Nonlinearity

The standard 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.

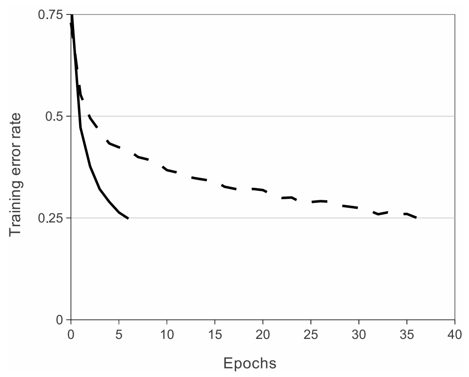

Figure 1: A four-layer convolutional neural network with ReLUs **(solid line)** reaches a 25% training error rate on CIFAR-10 six times faster than an equivalent network with $\tanh$ neurons **(dashed line)**. The learning rates for each network were chosen independently to make training as fast as possible. 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.

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.

3.2 Training on Multiple GPUs

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.

The resultant architecture is somewhat similar to that of the "columnar" CNN employed by Cireşan 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].

[2]The one-GPU net actually has the same number of kernels as the two-GPU net in the final convolutional layer. This is because most of the net's parameters are in the first fully-connected layer, which takes the last convolutional layer as input. So to make the two nets have approximately the same number of parameters, we did not halve the size of the final convolutional layer (nor the fully-conneced layers which follow). Therefore this comparison is biased in favor of the one-GPU net, since it is bigger than "half the size" of the two-GPU net.

3.1 ReLU Nonlinearity

将神经元的输出f建模为其输入x函数的标准方法是使用$f(x) = \tanh(x)$或$f(x) = (1+e^{-x})^{-1}$。在使用梯度下降进行训练时，这些饱和非线性函数的速度远不如非饱和非线性函数$f(x) = \max(0,x)$。根据Nair和Hinton的研究[20]，我们将具有这种非线性特性的神经元称为线性整流单元（ReLUs）。采用ReLU的深度卷积神经网络训练速度比使用tanh单元的等效网络快数倍。图1展示了这一现象，图中显示了在CIFAR-10数据集上，某个特定四层卷积网络达到25%训练误差所需的迭代次数。该曲线图表明，若使用传统的饱和神经元模型，我们便无法在本研究中开展如此大规模神经网络的实验。

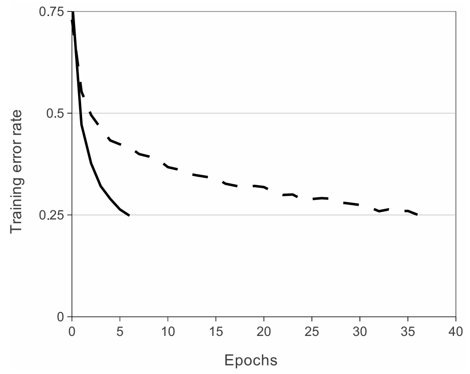

图1：一个带有ReLU激活函数（实线）的四层卷积神经网络在CIFAR-10数据集上达到25%训练错误率的速度，比使用tanh神经元（虚线）的同等网络快六倍。每个网络的学习率均经过独立选择以实现最快训练速度，未采用任何形式的正则化。此处展示的效果强度随网络架构而变化，但配备ReLU的网络始终比使用饱和神经元的同等网络快数倍学习。

我们并非首个考虑在CNN中替代传统神经元模型的人。例如，Jarrett等人[11]指出，在Caltech-101数据集上，非线性函数$f(x) = |\tanh(x)|$配合其提出的对比度归一化及局部平均池化方法效果显著。但该数据集的研究重点在于防止过拟合，因此他们观察到的现象与我们使用ReLU时发现的加速训练集拟合能力有所不同。更快的学习速度对大型数据集训练的大型模型性能具有重要影响。

3.2 多GPU训练

单个GTX 580 GPU仅有3GB内存，这限制了可在其上训练的网络的最大规模。事实证明，120万个训练样本足以训练那些因规模过大而无法在单个GPU上容纳的网络。因此我们将网络分布在两个GPU上。当前的GPU特别适合跨GPU并行化，因为它们能够直接读写彼此的内存，无需经过主机内存。我们采用的并行化方案本质上将一半的卷积核（或神经元）置于每个GPU上，并附加一个技巧：GPU仅在特定层进行通信。这意味着，例如第3层的卷积核接收来自第2层所有卷积核映射的输入，而第4层的卷积核仅接收位于同一GPU上的第3层卷积核映射的输入。连接模式的选择需要通过交叉验证来确定，但这使我们能够精确调整通信量，直至其达到计算量中可接受的比例。

最终的架构与Cireşan等人[5]采用的“柱状”CNN有些相似，不同之处在于我们的各列并非独立（见图2）。与在单个GPU上训练、每个卷积层核数减半的网络相比，该方案使我们的top-1和top-5错误率分别降低了1.7%和1.2%。双GPU网络的训练时间略短于单GPU网络[2]。

[2]The one-GPU net actually has the same number of kernels as the two-GPU net in the final convolutional layer. This is because most of the net's parameters are in the first fully-connected layer, which takes the last convolutional layer as input. So to make the two nets have approximately the same number of parameters, we did not halve the size of the final convolutional layer (nor the fully-conneced layers which follow). Therefore this comparison is biased in favor of the one-GPU net, since it is bigger than "half the size" of the two-GPU net.

3.3 Local Response Normalization

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

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

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 $k = 2$, $n = 5$, $\alpha = 10^{-4}$, and $\beta = 0.75$. We applied this normalization after applying the ReLU nonlinearity in certain layers (see Section 3.5).

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].

3.4 Overlapping Pooling

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.

3.5 Overall Architecture

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.

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.

The first convolutional layer filters the $224 \times 224 \times 3$ input image with 96 kernels of size $11 \times 11 \times 3$ with a stride of 4 pixels (this is the distance between the receptive field centers of neighboring

[3]We cannot describe this network in detail due to space constraints, but it is specified precisely by the code and parameter files provided here: http://code.google.com/p/cuda-convnet/.

3.3 局部响应归一化

ReLU具有一个理想的特性，即它们不需要输入归一化来防止饱和。如果至少有一些训练样本对ReLU产生正输入，那么该神经元就会发生学习。然而，我们仍然发现以下的局部归一化方案有助于泛化。记通过应用核i在位置(x, y)计算并应用ReLU非线性得到的神经元活动为$a^i_{x,y}$，则响应归一化后的活动$b^i_{x,y}$由表达式给出

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

其中求和遍历同一空间位置上“相邻”的核映射n，而N是该层中核的总数。核映射的顺序当然是任意的，并在训练开始前确定。这种响应归一化实现了一种侧向抑制形式，灵感来源于真实神经元中发现的类型，从而在使用不同核计算的神经元输出之间产生了对较大活动的竞争。常数k, n, α和β是超参数，其值通过验证集确定；我们使用了$k =$2、$n =$5、$\alpha =$10^{-4}和$\beta =$0.75。我们在某些层应用ReLU非线性后应用了这种归一化（参见第3.5节）。

该方案与Jarrett等人[11]的局部对比度归一化方案有相似之处，但我们的方法更应称为“亮度归一化”，因为我们并未减去平均激活值。响应归一化使我们的top-1和top-5错误率分别降低了1.4%和1.2%。我们还在CIFAR-10数据集上验证了该方案的有效性：一个四层CNN在未归一化时测试错误率为13%，归一化后降至11%[3]。

3.4 重叠池化

CNN中的池化层汇总了同一核映射中相邻神经元组的输出。传统上，相邻池化单元汇总的邻域互不重叠（例如[17, 11, 4]）。更准确地说，池化层可视为由间距为s像素的池化单元网格组成，每个单元汇总以池化单元位置为中心、尺寸为$z \times z$的邻域。若设$s = z$，则得到CNN常用的传统局部池化；若设$s < z$，则得到重叠池化。我们在整个网络中采用$s =$=2、$z =$=3的重叠池化方案。与输出尺寸相同的非重叠方案$s =$=2、, $z =$=2相比，该方案将top-1和top-5错误率分别降低0.4%和0.3%。在训练中我们通常观察到，采用重叠池化的模型较难出现过拟合现象。

3.5 整体架构

现在，我们准备描述我们CNN的整体架构。如图2所示，该网络包含八个带权重的层；前五层是卷积层，其余三层是全连接层。最后一个全连接层的输出被送入一个1000路的softmax分类器，该分类器产生1000个类别标签上的概率分布。我们的网络最大化多项式逻辑回归目标，这相当于在预测分布下，最大化训练样本正确标签的对数概率的平均值。

第二、第四和第五卷积层的核只与位于同一GPU上的前一层的核图相连（见图2）。第三卷积层的核则与第二层的所有核图相连。全连接层中的神经元与前一层的所有神经元相连。响应归一化层跟随在第一和第二卷积层之后。如3.4节所述的最大池化层，跟随在两个响应归一化层以及第五卷积层之后。ReLU非线性被应用于每个卷积层和全连接层的输出。

第一个卷积层使用96个大小为11×11×3的核，以4像素的步长（这是相邻感受野中心之间的距离）对224×224×3的输入图像进行滤波。

[3]We cannot describe this network in detail due to space constraints, but it is specified precisely by the code and parameter files provided here: http://code.google.com/p/cuda-convnet/.

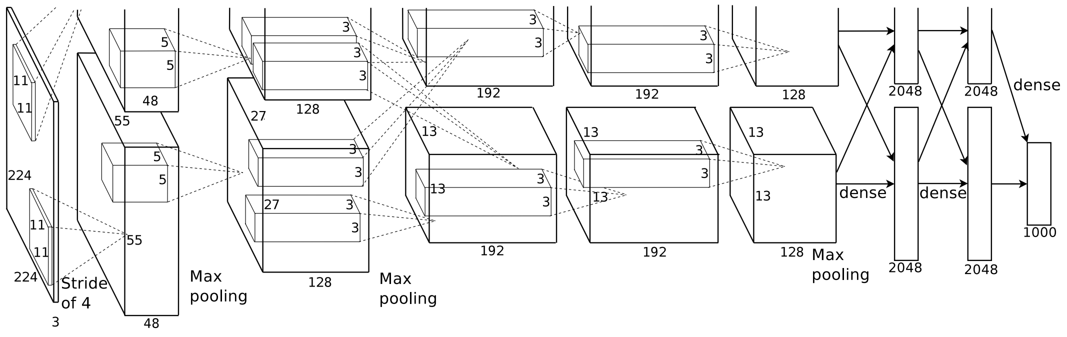

Figure 2: An illustration of the architecture of our CNN, explicitly showing the delineation of responsibilities between the two GPUs. One GPU runs the layer-parts at the top of the figure while the other runs the layer-parts at the bottom. The GPUs communicate only at certain layers. The network's input 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.

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 $5 \times 5 \times 48$. 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 $3 \times 3 \times 256$ connected to the (normalized, pooled) outputs of the second convolutional layer. The fourth convolutional layer has 384 kernels of size $3 \times 3 \times 192$, and the fifth convolutional layer has 256 kernels of size $3 \times 3 \times 192$. The fully-connected layers have 4096 neurons each.

4 Reducing Overfitting

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.

4.1 Data Augmentation

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.

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 interdependent. 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.

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,

[4]This is the reason why the input images in Figure 2 are $224 \times 224 \times 3$-dimensional.

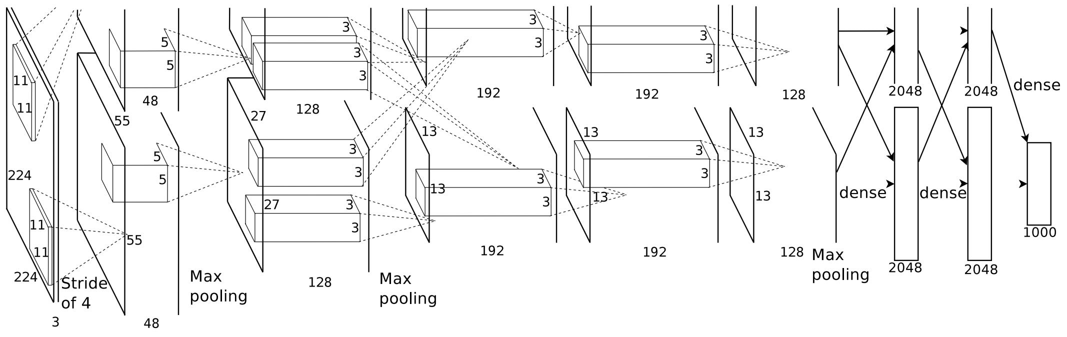

图2：我们CNN架构的示意图，明确展示了两块GPU之间的职责划分。一块GPU运行图中顶部的层部分，而另一块则运行底部的层部分。GPU仅在特定层进行通信。网络的输入维度为150,528，其余各层的神经元数量依次为253,440–186,624–64,896–64,896–43,264–4096–4096–1000。

核映射中的神经元）。第二卷积层以第一卷积层的（响应归一化与池化后的）输出作为输入，并用256个尺寸为5 × 5 × 48的核进行滤波。第三、第四和第五卷积层彼此相连，中间未插入任何池化或归一化层。第三卷积层拥有384个尺寸为3 × 3 × 256的核，与第二卷积层的（归一化、池化后）输出相连。第四卷积层包含384个尺寸为3 × 3 × 192的核，第五卷积层则包含256个尺寸为3 × 3 × 192的核。全连接层每层均包含4096个神经元。

4 减少过拟合

我们的神经网络架构拥有6000万个参数。尽管ILSVRC的1000个类别使得每个训练样本在从图像到标签的映射上施加了10比特的约束，但这仍不足以在避免严重过拟合的情况下学习如此多的参数。接下来，我们将阐述我们对抗过拟合的两种主要方法。

4.1 数据增强

减少图像数据过拟合最简单且最常用的方法，是通过标签保留变换（例如[25, 4, 5]）人为扩展数据集。我们采用两种不同的数据增强形式，这两种形式都能以极少的计算量从原始图像生成变换后的图像，因此无需将变换后的图像存储在磁盘上。在我们的实现中，变换图像由CPU端的Python代码生成，而GPU同时正在训练前一批图像。因此，这些数据增强方案实际上不产生额外计算开销。

数据增强的第一种形式包括生成图像平移和水平翻转。我们通过从256×256图像中提取随机的224×224图像块（及其水平翻转），并在这些提取的图像块上训练我们的网络来实现这一点。这使我们的训练集规模增加了2048倍，尽管由此产生的训练样本当然是高度相互依赖的。若不采用此方案，我们的网络会出现严重的过拟合，这将迫使我们使用更小的网络。在测试时，网络通过提取五个224×224图像块（四个角块和中心块）及其水平翻转（共十个图像块），并对网络softmax层在这十个图像块上的预测结果取平均值来进行预测。

第二种数据增强的形式包括改变训练图像中RGB通道的强度。具体来说，我们对整个ImageNet训练集中的RGB像素值集合进行主成分分析（PCA）。对每张训练图像，我们添加所找到的主成分的倍数，

4这 i　s the reason why the input images in Figure 2 are 224 × 224 × 3-dim维度的。

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 $I_{xy} = [I_{xy}^R, I_{xy}^G, I_{xy}^B]^T$ we add the following quantity:

$$[\mathbf{p}_1, \mathbf{p}_2, \mathbf{p}_3][\alpha_1\lambda_1, \alpha_2\lambda_2, \alpha_3\lambda_3]^T$$

where $\mathbf{p}_i$ 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%.

4.2 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.

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.

5 Details of learning

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

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

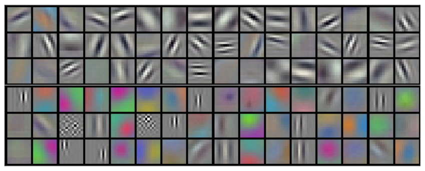
Figure 3: 96 convolutional kernels of size $11\times11\times3$ learned by the first convolutional layer on the $224\times224\times3$ input images. The top 48 kernels were learned on GPU 1 while the bottom 48 kernels were learned on GPU 2. See Section 6.1 for details.

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

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.

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

其大小与对应的特征值乘以一个从均值为零、标准差为0.1的高斯分布中抽取的随机变量成比例。因此，对每个RGB图像像素 $I_{xy} = [I_{xy}^R, I_{xy}^G, I_{xy}^B]^T$，我们添加以下量：

$$[\mathbf{p}_1, \mathbf{p}_2, \mathbf{p}_3][\alpha_1\lambda_1, \alpha_2\lambda_2, \alpha_3\lambda_3]^T$$

其中$\mathbf{p}_i$和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是前述随机变量。每个α_i仅针对特定训练图像的所有像素抽取一次，直到该图像再次用于训练时重新抽取。该方案近似捕捉了自然图像的一个重要特性，即物体身份对照明强度和颜色变化具有不变性。此方案使top-1错误率降低了超过1%。

4.2 丢弃法

结合多种不同模型的预测是减少测试误差的一种非常成功的方法[1, 3]，但对于已经需要数天训练时间的大型神经网络而言，这似乎过于昂贵。然而，有一种非常高效的模型组合变体，在训练期间仅需付出约两倍的成本。这种最近提出的技术称为“丢弃法”（dropout）[10]，其核心是以0.5的概率将每个隐藏神经元的输出置零。以这种方式被“丢弃”的神经元在前向传播中不产生作用，也不参与反向传播。因此，每次输入数据时，神经网络都会采样得到不同的架构，但所有这些架构共享权重。该技术减少了神经元间复杂的协同适应效应，因为单个神经元无法依赖特定其他神经元的存在。这迫使神经元必须学习更具鲁棒性的特征，这些特征需要与其他神经元的大量随机子集协同生效。在测试阶段，我们使用所有神经元，但将其输出乘以0.5，这是对指数级数量的丢弃网络所产生的预测分布进行几何平均的一种合理近似。

我们在图2的前两个全连接层中使用了dropout。如果不使用dropout，我们的网络会出现严重的过拟合。Dropout大约使收敛所需的迭代次数增加了一倍。

5 学习详情

我们使用随机梯度下降法训练模型，批大小为128个样本，动量为0.9，权重衰减为0.0005。我们发现这种微小的权重衰减对模型学习至关重要。换言之，此处的权重衰减不仅是正则化手段：它降低了模型的训练误差。权重w的更新规则为

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

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

其中 i 是迭代索引，v 是动量变量，ϵ 是学习率，$\left\langle \frac{\partial L}{\partial w}\Big|_{w_i} \right\rangle_{D_i}$ 是目标函数对 w 的导数在第 i 个批次 D_i 上的平均值，该导数在 w_i 处计算。

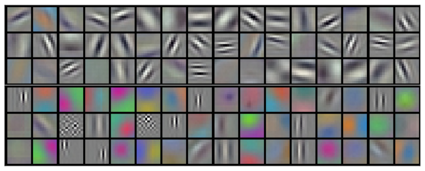
图3：在224×224×3输入图像上，由第一个卷积层学习到的96个大小为11×11×3的卷积核。顶部48个核在GPU 1上学习，底部48个核在GPU 2上学习。详见第6.1节。

我们将每层的权重初始化为标准差为0.01的零均值高斯分布。在第二、第四和第五卷积层以及全连接隐藏层中，我们将神经元偏置初始化为常数1。这种初始化通过为ReLU提供正输入，加速了学习的早期阶段。其余层的神经元偏置则初始化为常数0。

我们对所有层使用了相同的学习率，并在整个训练过程中手动调整。遵循的启发式方法是，当验证错误率在当前学习率下停止改善时，将学习率除以10。学习率初始设定为0.01，

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.

6 Results

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%**[5]. 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].

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.

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].

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.

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].

6.1 Qualitative Evaluations

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).

[5]The error rates without averaging predictions over ten patches as described in Section 4.1 are 39.0% and 18.3%.

在终止前减少了三次。我们使用包含120万张图像的训练集对网络进行了大约90个周期的训练，这在两个NVIDIA GTX 580 3GB GPU上花费了五到六天时间。

6 结果

我们在ILSVRC-2010上的结果总结在表1中。我们的网络实现了37.5%的top-1错误率和17.0%的top-5测试集错误率[5]。ILSVRC-2010竞赛期间的最佳性能为47.1%和28.2%，该方法是基于在不同特征上训练的六个稀疏编码模型的预测结果进行平均得出的[2]；此后已发表的最佳结果为45.7%和25.7%，该方法是对两种密集采样特征计算的Fisher向量（FV）训练的两个分类器的预测结果进行平均得出的[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%**

表1：ILSVRC-2010测试集结果对比。*italics*中为其他方法取得的最佳结果。

我们也将我们的模型提交至ILSVRC-2012竞赛，并在表2中汇报了结果。由于ILSVRC-2012测试集的标签未公开，我们无法对所有尝试的模型报告测试错误率。在本段余下部分，我们将验证集和测试集错误率互换使用，因为根据我们的经验，两者的差异不超过0.1%（见表2）。本文描述的CNN取得了18.2%的top-5错误率。通过平均预测五个相似CNN的平均错误率为16.4%。训练一个在最后一个池化层上额外增加第六个卷积层的CNN，对整个ImageNet 2011年秋季数据集（1500万张图像，2.2万个类别）进行分类，再在ILSVRC-2012上对其进行“微调”，所得错误率为16.6%。将两个基于整个2011年秋季数据集预训练的CNN与前述五个CNN的预测结果进行平均，错误率降至15.3%。竞赛中第二佳的成绩是通过对基于不同类型密集采样特征计算的FVs训练出的多个分类器预测结果进行平均而实现的，其错误率为26.2%[7]。

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%**

表2：ILSVRC-2012验证集和测试集错误率对比。*italics*为其他方法取得的最佳结果。带星号*的模型经过“预训练”以分类整个ImageNet 2011秋季发布版本。详见第6节。

最后，我们还在2009年秋季版本的ImageNet数据集上报告了错误率，该数据集包含10,184个类别和890万张图像。在此数据集上，我们遵循文献中的惯例，使用一半图像进行训练，另一半进行测试。由于没有既定的测试集，我们的划分必然与先前作者使用的划分不同，但这并未显著影响结果。我们在该数据集上的top-1和top-5错误率分别为67.4%和40.9%，由上述网络实现，但在最后一个池化层上额外增加了一个第六卷积层。该数据集上已发表的最佳结果为78.1%和60.9%[19]。

6.1 定性评估

图3展示了网络两个数据连接层学习到的卷积核。该网络已习得多种频率与方向选择性核，以及各类彩色斑块。值得注意的是两个GPU表现出的专业化特性，这是3.5节所述受限连接机制的结果。GPU 1上的核基本与颜色无关，而GPU 2上的核则主要具有颜色特异性。此类专业化现象在每次训练中都会出现，且不受任何特定随机权重初始化影响（除GPU编号可能重新分配外）。

[5]The error rates without averaging predictions over ten patches as described in Section 4.1 are 39.0% and 18.3%.

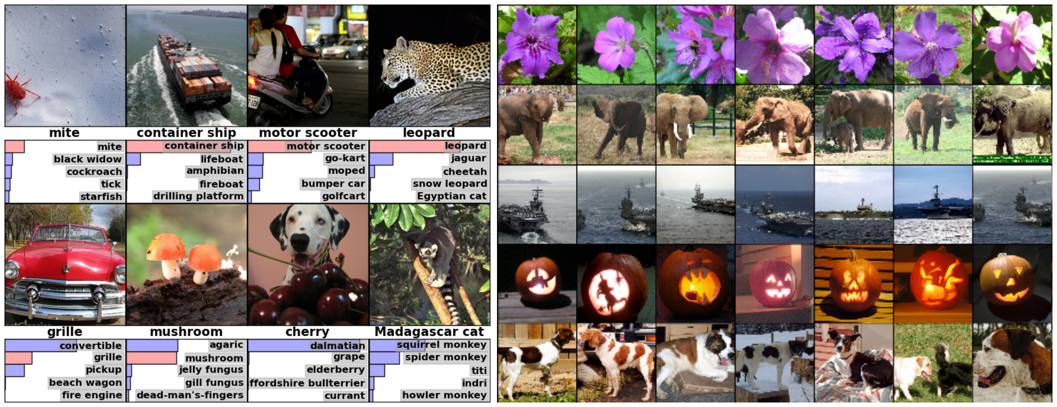

Figure 4: **(Left)** Eight ILSVRC-2010 test images and the five labels considered most probable by our model. The correct label is written under each image, and the probability assigned to the correct label is also shown with a red bar (if it happens to be in the top 5). **(Right)** Five ILSVRC-2010 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.

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.

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.

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.

7 Discussion

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.

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.

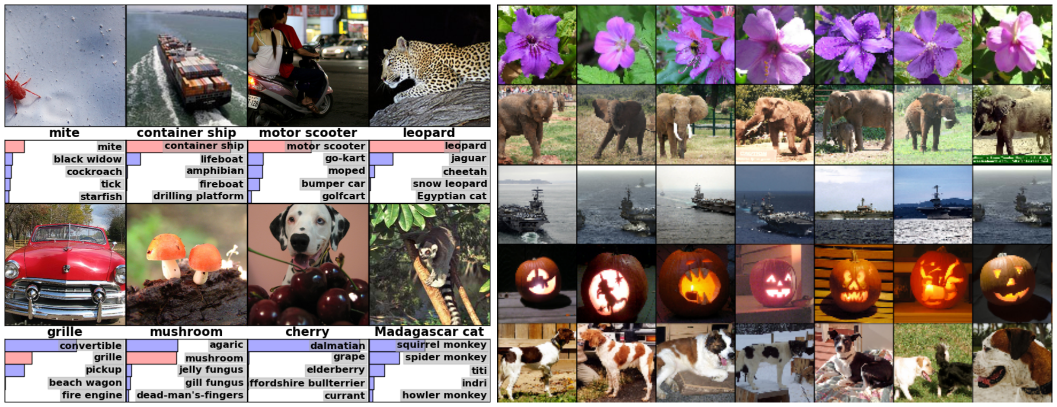

图4：（左）八张ILSVRC-2010测试图像及我们的模型认为最可能的五个标签。每张图像下方标注了正确标签，同时以红色条形图显示模型分配给正确标签的概率（若该标签恰好位于前五名）。（右）第一列显示五张ILSVRC-2010测试图像，其余列展示六张训练图像——这些训练图像在最后一层隐藏层中生成的特征向量，与测试图像特征向量之间的欧几里得距离最小。

在图4的左面板中，我们通过计算网络对八张测试图像的前5位预测，定性评估了网络所学到的内容。请注意，即使是偏离中心的物体，例如左上角的螨虫，也能被网络识别。大多数前5位的标签看起来是合理的。例如，对于豹子，只有其他类型的猫科动物被认为是合理的标签。在某些情况下（格栅、樱桃），照片的预期焦点确实存在歧义。

另一种探究网络视觉知识的方法是考虑图像在最后一个4096维隐藏层上引发的特征激活。如果两幅图像产生的特征激活向量具有较小的欧几里得距离，我们可以说神经网络的较高层级认为它们是相似的。图4展示了测试集中的五幅图像，以及根据此度量标准，训练集中与它们各自最相似的六幅图像。请注意，在像素级别上，检索到的训练图像与第一列中的查询图像在L2距离上通常并不接近。例如，检索到的狗和大象以多种姿态出现。我们在补充材料中展示了更多测试图像的结果。

通过计算两个4096维实值向量之间的欧几里得距离来衡量相似性效率较低，但可以通过训练自编码器将这些向量压缩为短二进制码来实现高效计算。相比将自编码器直接应用于原始像素的方法[14]，这种方法应能产生更好的图像检索效果——原始像素方法未利用图像标签，容易检索出边缘模式相似但语义未必相关的图像。

7 讨论

我们的结果表明，一个大型、深层的卷积神经网络能够在极具挑战性的数据集上，仅通过纯监督学习就取得破纪录的结果。值得注意的是，如果移除单个卷积层，我们网络的性能就会下降。例如，移除任何中间层都会导致网络在top-1性能上损失约2%。因此，深度对于实现我们的结果确实至关重要。

为了简化实验，我们并未使用任何无监督预训练，尽管我们预期这会有所帮助，特别是当我们获得足够的计算能力，能够在未相应增加标注数据量的情况下显著扩大网络规模时。迄今为止，随着我们扩大网络规模并延长训练时间，结果已有所改善，但要匹配人类视觉系统的颞下通路，我们仍需跨越多个数量级的差距。最终，我们希望将超大规模深度卷积网络应用于视频序列，其中时间结构能提供静态图像中缺失或远不明显的重要信息。

References

[1] R.M. Bell and Y. Koren. Lessons from the netflix prize challenge. *ACM SIGKDD Explorations Newsletter*, 9(2):75–79, 2007.

[2] A. Berg, J. Deng, and L. Fei-Fei. Large scale visual recognition challenge 2010. www.image-net.org/challenges. 2010.

[3] L. Breiman. Random forests. *Machine learning*, 45(1):5–32, 2001.

[4] D. Cireşan, U. Meier, and J. Schmidhuber. Multi-column deep neural networks for image classification. *Arxiv preprint arXiv:1202.2745*, 2012.

[5] D.C. Cireşan, U. Meier, J. Masci, L.M. Gambardella, and J. Schmidhuber. High-performance neural networks for visual object classification. *Arxiv preprint arXiv:1102.0183*, 2011.

[6] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. ImageNet: A Large-Scale Hierarchical Image Database. In *CVPR09*, 2009.

[7] J. Deng, A. Berg, S. Satheesh, H. Su, A. Khosla, and L. Fei-Fei. *ILSVRC-2012*, 2012. URL `http://www.image-net.org/challenges/LSVRC/2012/`.

[8] L. Fei-Fei, R. Fergus, and P. Perona. Learning generative visual models from few training examples: An incremental bayesian approach tested on 101 object categories. *Computer Vision and Image Understanding*, 106(1):59–70, 2007.

[9] G. Griffin, A. Holub, and P. Perona. Caltech-256 object category dataset. Technical Report 7694, California Institute of Technology, 2007. URL `http://authors.library.caltech.edu/7694`.

[10] G.E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, and R.R. Salakhutdinov. Improving neural networks by preventing co-adaptation of feature detectors. *arXiv preprint arXiv:1207.0580*, 2012.

[11] K. Jarrett, K. Kavukcuoglu, M. A. Ranzato, and Y. LeCun. What is the best multi-stage architecture for object recognition? In *International Conference on Computer Vision*, pages 2146–2153. IEEE, 2009.

[12] A. Krizhevsky. Learning multiple layers of features from tiny images. Master's thesis, Department of Computer Science, University of Toronto, 2009.

[13] A. Krizhevsky. Convolutional deep belief networks on cifar-10. *Unpublished manuscript*, 2010.

[14] A. Krizhevsky and G.E. Hinton. Using very deep autoencoders for content-based image retrieval. In *ESANN*, 2011.

[15] Y. Le Cun, B. Boser, J.S. Denker, D. Henderson, R.E. Howard, W. Hubbard, L.D. Jackel, et al. Handwritten digit recognition with a back-propagation network. In *Advances in neural information processing systems*, 1990.

[16] Y. LeCun, F.J. Huang, and L. Bottou. Learning methods for generic object recognition with invariance to pose and lighting. In *Computer Vision and Pattern Recognition, 2004. CVPR 2004. Proceedings of the 2004 IEEE Computer Society Conference on*, volume 2, pages II–97. IEEE, 2004.

[17] Y. LeCun, K. Kavukcuoglu, and C. Farabet. Convolutional networks and applications in vision. In *Circuits and Systems (ISCAS), Proceedings of 2010 IEEE International Symposium on*, pages 253–256. IEEE, 2010.

[18] H. Lee, R. Grosse, R. Ranganath, and A.Y. Ng. Convolutional deep belief networks for scalable unsupervised learning of hierarchical representations. In *Proceedings of the 26th Annual International Conference on Machine Learning*, pages 609–616. ACM, 2009.

[19] T. Mensink, J. Verbeek, F. Perronnin, and G. Csurka. Metric Learning for Large Scale Image Classification: Generalizing to New Classes at Near-Zero Cost. In *ECCV - European Conference on Computer Vision*, Florence, Italy, October 2012.

[20] V. Nair and G. E. Hinton. Rectified linear units improve restricted boltzmann machines. In *Proc. 27th International Conference on Machine Learning*, 2010.

[21] N. Pinto, D.D. Cox, and J.J. DiCarlo. Why is real-world visual object recognition hard? *PLoS computational biology*, 4(1):e27, 2008.

[22] N. Pinto, D. Doukhan, J.J. DiCarlo, and D.D. Cox. A high-throughput screening approach to discovering good forms of biologically inspired visual representation. *PLoS computational biology*, 5(11):e1000579, 2009.

[23] B.C. Russell, A. Torralba, K.P. Murphy, and W.T. Freeman. Labelme: a database and web-based tool for image annotation. *International journal of computer vision*, 77(1):157–173, 2008.

[24] J. Sánchez and F. Perronnin. High-dimensional signature compression for large-scale image classification. In *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*, pages 1665–1672. IEEE, 2011.

[25] P.Y. Simard, D. Steinkraus, and J.C. Platt. Best practices for convolutional neural networks applied to visual document analysis. In *Proceedings of the Seventh International Conference on Document Analysis and Recognition*, volume 2, pages 958–962, 2003.

[26] S.C. Turaga, J.F. Murray, V. Jain, F. Roth, M. Helmstaedter, K. Briggman, W. Denk, and H.S. Seung. Convolutional networks can learn to generate affinity graphs for image segmentation. *Neural Computation*, 22(2):511–538, 2010.

参考文献

[1] R.M. Bell 与 Y. Koren。Netflix 大奖挑战赛的启示。*ACM SIGKDD Explorations Newsletter*，第9卷第2期，第75−79页，2007年。[2] A. Berg、J. Deng 与 L. Fei-Fei。2010年大规模视觉识别挑战赛。www.image-net.org/challenges。2010年。[3] L. Breiman。随机森林。*Machine learning*，第45卷第1期，第5−32页，2001年。[4] D. Cireşan、U. Meier 与 J. Schmidhuber。用于图像分类的多列深度神经网络。*Arxiv preprint arXiv:1202.2745*，2012年。[5] D.C. Cireşan、U. Meier、J. Masci、L.M. Gambardella 与 J. Schmidhuber。用于视觉对象分类的高性能神经网络。*Arxiv preprint arXiv:1102.0183*，2011年。[6] J. Deng、W. Dong、R. Socher、L.-J. Li、K. Li 与 L. Fei-Fei。ImageNet：一个大规模分层图像数据库。收录于 *CVPR09*，2009年。[7] J. Deng、A. Berg、S. Satheesh、H. Su、A. Khosla 与 L. Fei-Fei。*ILSVRC-2012*，2012年。网址 http://www.image-net.org/challenges/LSVRC/2012/。[8] L. Fei-Fei、R. Fergus 与 P. Perona。从少量训练样本中学习生成式视觉模型：在101个对象类别上测试的增量贝叶斯方法。*Computer Vision and Image Understand- ing*，第106卷第1期，第59−70页，2007年。[9] G. Griffin、A. Holub 与 P. Perona。Caltech-256 对象类别数据集。技术报告 7694，加州理工学院，2007年。网址 http://authors.library.caltech.edu/7694。[10] G.E. Hinton、N. Srivastava、A. Krizhevsky、I. Sutskever 与 R.R. Salakhutdinov。通过防止特征检测器的共适应来改进神经网络。*arXiv preprint arXiv:1207.0580*，2012年。[11] K. Jarrett、K. Kavukcuoglu、M. A. Ranzato 与 Y. LeCun。什么是最佳的多阶段物体识别架构？收录于 *International Conference on Computer Vision*，第2146−2153页。IEEE，2009年。[12] A. Krizhevsky。从微小图像中学习多层特征。硕士论文，多伦多大学计算机科学系，2009年。[13] A. Krizhevsky。在 CIFAR-10 上的卷积深度置信网络。*Unpublished manuscript*，2010年。[14] A. Krizhevsky 与 G.E. Hinton。使用非常深的自动编码器进行基于内容的图像检索。收录于 *ESANN*，2011年。[15] Y. Le Cun、B. Boser、J.S. Denker、D. Henderson、R.E. Howard、W. Hubbard、L.D. Jackel 等。使用反向传播网络进行手写数字识别。收录于 *Advances in neural information processing systems*，1990年。[16] Y. LeCun、F.J. Huang 与 L. Bottou。对姿态和光照具有不变性的通用物体识别学习方法。收录于 *Computer Vision and Pattern Recognition, 2004. CVPR 2004. Proceedings of the 2004 IEEE Computer Society Conference on*，第2卷，第 II−97 页。IEEE，2004年。[17] Y. LeCun、K. Kavukcuoglu 与 C. Farabet。卷积网络及其在视觉中的应用。收录于 *Circuits and Systems (ISCAS), Proceedings of 2010 IEEE International Symposium on*，第253−256页。IEEE，2010年。[18] H. Lee、R. Grosse、R. Ranganath 与 A.Y. Ng。用于可扩展无监督分层表示学习的卷积深度置信网络。收录于 *Proceedings of the 26th Annual International Conference on Machine Learning*，第609−616页。ACM，2009年。[19] T. Mensink、J. Verbeek、F. Perronnin 与 G. Csurka。大规模图像分类的度量学习：以接近零成本泛化到新类别。收录于 *ECCV - European Conference on Computer Vision*，意大利佛罗伦萨，2012年10月。[20] V. Nair 与 G. E. Hinton。修正线性单元改进受限玻尔兹曼机。收录于 *Proc. 27th International Conference on Machine Learning*，2010年。[21] N. Pinto、D.D. Cox 与 J.J. DiCarlo。为什么真实世界的视觉物体识别很困难？*PLoS computa- tional biology*，第4卷第1期，第e27页，2008年。[22] N. Pinto、D. Doukhan、J.J. DiCarlo 与 D.D. Cox。一种通过高通量筛选发现良好生物启发视觉表示形式的方法。*PLoS computational biology*，第5卷第11期，第e1000579页，2009年。[23] B.C. Russell、A. Torralba、K.P. Murphy 与 W.T. Freeman。Labelme：一个图像标注数据库和基于网络的工具。*International journal of computer vision*，第77卷第1期，第157−173页，2008年。[24] J. Sánchez 与 F. Perronnin。用于大规模图像分类的高维签名压缩。收录于 *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*，第1665−1672页。IEEE，2011年。[25] P.Y. Simard、D. Steinkraus 与 J.C. Platt。应用于视觉文档分析的卷积神经网络最佳实践。收录于 *Proceedings of the Seventh International Conference on Document Analysis and Recognition*，第2卷，第958−962页，2003年。[26] S.C. Turaga、J.F. Murray、V. Jain、F. Roth、M. Helmstaedter、K. Briggman、W. Denk 与 H.S. Seung。卷积网络可以学习生成用于图像分割的亲和力图。*Neural Computation*，第22卷第2期，第511−538页，2010年。