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.

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为 1000 个不同的类别。在测试集上，我们达到了 37.5%和 17.0%的 top-1 和 top-5 错误率，这比之前的 state-of-the-art 要好得多。该神经网络拥有 6000 万个参数和 650,000 个神经元，由五个卷积层（其中一些卷积层后面是最大池化层），三个全连接层以及最后的 1000 维的 softmax 层组成。为了加快训练速度，我们使用了非饱和神经元和卷积运算的非常高效的 GPU 实现。为了减少全连接层的过度拟合，我们采用了最近提出的一种称为“dropout”的正则化方法，该方法被证明非常有效。我们还用模型的一个变体参加了 ILSVRC-2012 竞赛，并在测试集上取得了 15.3%的 top-5 错误率，而第二名的错误率为 26.2%。（**我很疑惑为什么没写 2012 年的 top-1 错误率，难道是因为结果不好吗？后来在 Section 2 中找到了答案，因为 10 年公布了测试集，12 年没有公布测试集，只能提交给系统评判，返回 top-5 错误率**）

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

机器学习是当前的物体识别的必要方法。为了提高它们的性能，我们可以收集更大的数据集，学习更强大的模型，并使用更好的技术来防止过度拟合（**在过去，方向是研究使用各种正则方法来阻止过拟合；现在，新的方向是似乎正则化没那么重要，只要设计好神经网络架构即可**）。直到最近，标记图像的数据集还相对较小——大约有数万

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 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.

张图像（例如 NORB [16]、Caltech-101/256 [8, 9]和 CIFAR-10/100 [12]）。使用这种大小的数据集可以很好地解决简单的识别任务，特别是如果它们通过保留标签的转换进行增强。例如，MNIST 数字识别任务的当前最佳错误率 (<0.3%) 接近人类的表现[4]。但是现实环境中的物体表现出相当大的可变性，因此要学会识别它们，就必须使用更大的训练集。事实上，小图像数据集的缺点已得到广泛认可（例如，Pinto 等人[21]），但直到最近才可能收集具有数百万张图像的标记数据集。新的更大的数据集包括由数十万张完全分割的图像组成的 LabelMe [23]和由超过 22000 个类别的超过 1500 万张标记的高分辨率图像组成的 ImageNet [6]。（**吹一波自己用的 ImageNet**）

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.

要从数百万张图像中识别数千个类别，我们需要一个具有很强的学习能力的模型。然而，对象识别任务的巨大复杂性意味着即使像 ImageNet 这样大的数据集也无法解决这个问题，因此我们的模型也应该有很多先验知识来补偿所有我们没有的数据。卷积神经网络 (CNN) 构成了这样一类模型 [16、11、13、18、15、22、26]。（**当时主流的模型并不是 CNN，但是这篇文章对它们只字不提，只是说自己的 CNN，格局小了**）它们的容量可以通过改变它们的深度和广度来控制，并且它们还对图像的性质（即统计数据的平稳性和像素依赖性的局部性）做出了强有力且基本正确的假设。因此，与具有类似大小层的标准前馈神经网络相比，CNN 具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只是稍微差一点。（**吹一波自己用的 CNN**）

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,

尽管 CNN 具有吸引人的品质，并且尽管它们的本地架构相对高效，但将它们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 2D 卷积实现相结合，足以促进有

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.

趣的大型 CNN 的训练，并且最近的数据集（例如 ImageNet）包含足够的标记示例来训练此类模型而不会出现严重的过度拟合。

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[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.

本文的具体贡献如下：我们在 ILSVRC-2010 和 ILSVRC-2012 比赛 [2] 中使用 ImageNet 的子集的数据训练了迄今为止最大的卷积神经网络之一，并取得了迄今为止在这些数据集上报道的最佳结果（**效果很好**）。我们编写了一个高度优化的 GPU 实现，用于 2D 卷积和训练卷积神经网络中固有的所有其他操作，我们将其公开[1]（**GPU 编程**）。我们的网络包含许多新的和不寻常的特征，可以提高其性能并减少其训练时间，详见第 3 节（**新的和不寻常的特征**）。我们网络的规模使得过度拟合成为一个严重问题，即使有 120 万个标记的训练示例，因此我们使用了几个防止过拟合的有效技术，在第 4 节中描述（**过拟合**）。我们的最终网络包含五个卷积层和三个全连接层，网络的深度似乎很重要：我们发现去除任何卷积层（每个卷积层包含不超过 1 %的模型参数）导致性能下降（**深度**）。

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，网络的大小主要受限于当前 GPU 上可用的内存量以及我们愿意容忍的训练时间。（**受限于 memory、time**）我们的网络需要五到六天的时间在两个 GTX 580 3GB GPU 上进行训练。我们所有的实验都表明，只需等待更快的 GPU 和更大的数据集可用，就可以改善我们的结果（**需要 faster GPU 和 bigger datasets**）。

2 The Dataset

ImageNet is a dataset of over 15 million

ImageNet 是一个包含超过1500万张有

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.

标签的的高分辨率图像的数据集，有大约 22,000 个类别。这些图像是从网络上收集的，并由人工标注者使用亚马逊的 Mechanical Turk 众包工具进行标记。从 2010 年开始，作为 Pascal Visual Object Challenge 的一部分，一年一度的竞赛称为 ImageNet Large-Scale Visual Recognition Challenge (ILSVRC)。ILSVRC 使用 ImageNet 的一个子集，在 1000 个类别中的每个类别中包含大约 1000 张图像。总共有大约 120 万张训练图像、50000 张验证图像和 150000 张测试图像。

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 是唯一公布测试集标签的 ILSVRC 版本，因此这是我们执行大部分实验的版本。由于我们还用我们的模型参加了 ILSVRC-2012 竞赛，因此在第 6 节中，我们也报告了此版本数据集的结果，其中测试集标签不公布。在 ImageNet 上，通常会报告两个错误率：top-1 和 top-5，其中 top-5 错误率是正确标签不在模型认为最可能的五个标签中的测试图像占全部测试图像的比例（**一张图像输入网络，最终经过 softmax 得到 1000 个概率，最大的五个概率对应的标签不包含正确的标签**）。

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 由不同的分辨率的图像组成，而我们的系统需要恒定的输入维度。因此，我们将图像下采样到固定分辨率256×256。给定一个矩形图像，我们首先重新缩放图像，使短边的长度为 256，然后从结果图像中裁剪出中心的256×256块。除了从每个像素中减去训练集上的平均部分外，我们没有以任何其他方式对图像进行预处理。因此，我们在像素的（中心）原始 RGB 值上训练了我们的网络（**没有抽取任何的特征，直接是在原始的像素上做训练，虽然作者当时并没有将此处作为卖点，但此处却催生了 end-to-end 的架构，即直接将原始的图片或文本作为网络的输入，简单有效**）。

3 The Architecture

The architecture of our network is summarized in Figure 2. It contains eight

图 2 总结了我们网络的架构。它包含八个学习层——五个卷积层和三个全连接层。

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.

下面，我们将描述我们网络架构的一些新颖或不寻常的功能。第 3.1-3.4 节根据我们对其重要性的估计进行排序，最重要的在前。

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.

将神经元的输出f建模为其输入x的函数的标准方法是$f(x) = \tanh(x)$或$f(x) = (1 + e^{-x})^{-1}$。使用就梯度下降的训练时间而言，这些饱和非线性比非饱和非线性$f(x) = \max(0, x)$慢得多。继Nair和Hinton [20]（**具体 ReLU 为什么快，去这篇论文里找原因**）之后，我们将具有这种非线性的神经元称为整流线性单元 (ReLU)。带有ReLU的深度卷积神经网络的训练速度比带有tanh 单元的同等网络快几倍。（**现在来看，ReLU 也没有快很多，但是 ReLU 更简单，简单就是胜利**）这在图 1 中得到了证明，它显示了在 CIFAR-10 数据集上针对特定的四层卷积网络达到 25% 训练误差所需的迭代次数。该图表明，如果我们使用传统的饱和神经元模型，我们将无法在这项工作中使用如此大的神经网络进行实验。

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.

我们并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]声称非线性的$f(x) = |\tanh(x)|$在 Caltech-101 数据集上进行对比度归一化和局部平均池化后效果特别好。然而，在这个数据集上，主要关注的是防止过度拟合，因此他们观察到的效果与我们在使用 ReLU 时报告的加速拟合训练集的能力不同。更快地学习对在大型数据集上训练的大型模型的性能有很大影响。（**与之前这方面类似的工作做比较**）

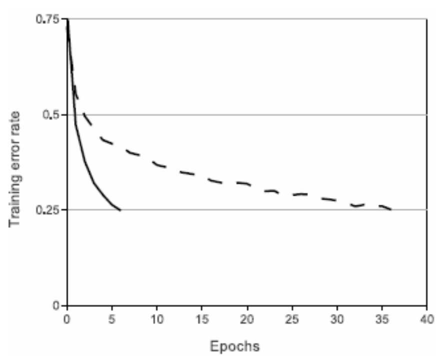


Figure 1: A four-layer convolutional neural network with ReLUs (**solid line**) reaches a 25% training error rate on CIFAR-10 six times faster than an equivalent network with tanh neurons (**dashed line**). The learning rates for each network were chosen independently to make training as fast as possible. 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：带有 ReLU 的四层卷积神经网络（**实线**）在 CIFAR-10 上达到 25%的训练错误率，比带有 tanh 神经元的等效网络（**虚线**）快六倍。每个网络的学习率都是独立选择的，以使训练尽可能快。没有采用任何形式的正则化。这里展示的效果的大小因网络架构而异，但使用 ReLU 的网络始终比使用饱和神经元的网络学习速度快几倍（**只写了 ReLU 快，没写 ReLU 为什么快，其实 ReLU 快是因为它不会像其它两个激活函数一样在梯度下降时陷入一个饱和区**）。

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 fraction of the amount of computation.

单个 GTX 580 GPU 只有 3GB 的内存，这限制了可以在其上训练的网络的最大规模。事实证明，120 万个训练示例足以训练太大而无法在一个 GPU 上安装的网络。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取和写入彼此的内存，而无需通过主机内存。我们采用的并行化方案本质上是将一半的内核（或神经元）放在每个 GPU 上，还有一个技巧：GPU 仅在某些层中进行通信。这意味着，例如，第 3 层的内核从第 2 层的所有内核映射中获取输入。然而，第 4 层中的内核仅从位于同一 GPU 上的第 3 层内核映射中获取输入。选择连接模式是交叉验证的一个问题，但这使我们能够精确地调整通信量，直到它是计算量的一个可接受的部分（**3.2 是具体的复杂的工程上的细节，与整体的方法无关**）。

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].

由此产生的架构有点类似于 Ciresan [5] 等人采用的“柱状”CNN 的架构，但是我们的列不是独立的（见图 2）。与在一个 GPU 上训练的每个卷积层中内核数量减半的网络相比，该方案分别将我们的 top-1 和 top-5 错误率降低了 1.7%和 1.2%。双 GPU 网络的训练时间比单 GPU 网络稍短 [2]。

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 $\mathrm{a}^{i}_{\mathrm{x,y}}$ the activity of a neuron computed by applying kernel i at position $(\mathrm{x,y})$ and then applying the ReLU nonlinearity, the response-normalized activity $\mathrm{b}^{i}_{\mathrm{x,y}}$ is given by the expression

ReLU 具有理想的特性，即它们不需要输入归一化来防止它们饱和。如果至少有一些训练样例对 ReLU 产生了正输入，那么学习就会发生在那个神经元中。但是，我们仍然发现以下局部归一化方案有助于泛化。用$\mathrm{a}^{i}_{\mathrm{x,y}}$表示通过在位置 (x,y) 应用内核 i 然后应用 ReLU 非线性计算的神经元变量，响应归一化变量$\mathrm{b}^{i}_{\mathrm{x,y}}$由表达式给出

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

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 $\mathrm{k}, \mathrm{n}, \alpha$ and β are hyper-parameters whose values are determined using a validation set; we used $\mathrm{k} = 2, \mathrm{n} = 5, \alpha = 10^{-4}$ and $\beta = 0.75$. We applied this normalization after applying the ReLU nonlinearity in certain layers (see Section 3.5).

其中求和在相同空间位置的 n 个“相邻”内核映射上运行，N 是层中内核的总数。内核映射的排序当然是任意的，并且在训练开始之前就已确定。这种响应归一化实现了一种受真实神经元类型启发的侧向抑制形式，在使用不同内核计算的神经元输出之间创建大型活动的竞争。常数k, n, α和β是超参数，其值由验证集确定；我们使用$\mathrm{k} = 2, \mathrm{n} = 5, \alpha = 10^{-4}$和$\beta = 0.75$。我们在某些层中应用了 ReLU 非线性之后应用了这种归一化（详见第 3.5 节）(**这是一个归一化的方法，能够避免饱和，但很少有人使用这种方法，现在有更好的归一化方法**)。

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].

该方案与 Jarrett 等人[11]的局部对比度归一化方案有一些相似之处，但我们将其称为“brightness normalization”更正确，因为我们没有减去平均活动。 响应归一化将我们的 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：四层 CNN 在没有归一化的情况下达到了 13%的测试集错误率，在归一化的情况下达到了 11%。

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.

CNN 的池化层汇总了同一内核映射中相邻神经元组的输出。传统上，相邻池化单元总结的邻域不重叠（例如，[17, 11, 4]）。更准确地说，一个池化层可以被认为是由一个间隔 s 个像素的池化单元网格组成，每个网格汇总了一个以池化单元位置为中心的大小为$z \times z$的邻域。如果我们设置$s = z$，我们将获得 CNN 中常用的传统局部池化。如果我们设置$s < z$，我们将获得重叠池化。这是我们在整个网络中使用的，$s = 2$和$z = 3$。与产生等效维度输出的非重叠方案 s=2 相比，该方案分别将 top-1 和 top-5 错误率降低了 0.4%和 0.3%。我们在训练期间观察到具有重叠池化的模型更不容易过拟合。**（对传统的不重叠的池化改进为重叠的）**

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 fullyconnected. The output of the last fully-connected layer is fed to a 1000-way softmax which produces a

现在我们已经准备好描述我们的 CNN 的整体架构。如图 2 所示，网络包含八个带有权重的层；前五个是卷积层，剩余三个是全连接层。最后一个全连接层的输出被输入到一个 1000 维的 softmax，它产生 1000 个类别标签的分布。（**网络整体架构**）我们的网络最大化多项逻辑回归目标，这相当于最

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.

大化预测分布下正确标签的对数概率的训练案例的平均值（**交叉熵损失函数**）。

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 fullyconnected 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.

第二、第四和第五卷积层的内核仅连接到前一层中位于同一 GPU 上的内核映射（见图 2）。第三个卷积层的内核连接到第二层的所有 GPU 上的内核映射。全连接层中的神经元连接到前一层中的所有神经元。响应归一化层跟在第一和第二个卷积层之后。3.4 节中描述的那种最大池化层，跟在响应归一化层和第五个卷积层之后。非线性的 ReLU 应用于每个卷积层和全连接层的输出。（**讲了归一化层和池化层的位置以及两个 GPU 协同工作的细节**）

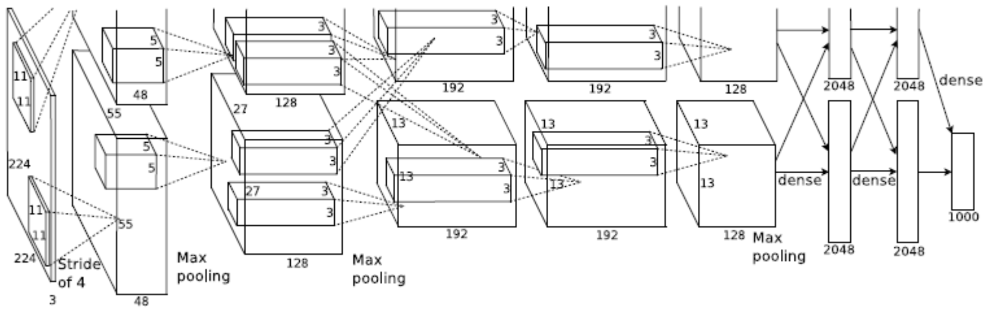


Figure 2: An illustration of the architecture of our CNN, explicitly showing the delineation of responsibilities between the two GPUs. One GPU runs the layer-parts at the top of the figure while the other runs the layer-parts at the bottom. The GPUs communicate only at certain layers. The network's input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

图 2：我们的 CNN 的架构图，明确显示了两个 GPU 之间的职责划分。一个 GPU 运行图形顶部的层部分，而另一个运行底部的层部分。GPU 之间仅在某些层进行通信。网络的输入为 150528 维，网络其余层的神经元数量为 253,440-186,624-64,896-64,896-43,264-4096-4096-1000。（**不难发现，瓶颈在前两个全连接层之间**）

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 neurons in a kernel map). The second convolutional layer takes as

第一个卷积层用 96 个大小为$11 \times 11 \times 3$的内核过滤$224 \times 224 \times 3$的输入图像，步长为 4 个像素（这是内核图中相邻神经元的感受野中心之间的距离）。第二个卷积层将第一个卷积层的（响应归一化和池化）输出作为输入，并用 256 个大小为$5 \times 5 \times$

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.

48的内核对其进行过滤。第三、第四和第五卷积层相互连接，中间没有任何池化或归一化层。第三个卷积层有 384 个大小为$3 \times 3 \times 256$的内核，连接到第二个卷积层的（归一化、池化）输出。第四个卷积层 384 个大小为$3 \times 3 \times 192$的内核，第五个卷积层有 256 个大小为$3 \times 3 \times 192$的内核。全连接层各有 4096 个神经元。（**从头到尾走了一遍网络，说清楚了中间的超参数**）

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.

我们的神经网络架构有 6000 万个参数。尽管 ILSVRC 的 1000 个类别使每个训练示例对从图像到标签的映射施加了 10 位的约束（**没看懂这句话**），但事实证明，这不足以学习如此多的参数而不会出现严重的过拟合。下面，我们将描述我们对抗过度拟合的两种主要方式。

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.

减少图像数据过度拟合的最简单和最常见的方法是使用标签保留转换（例如[25, 4, 5]）人为地扩大数据集。我们采用了两种不同形式的数据增强，这两种形式都允许以很少的计算从原始图像生成转换后的图像，因此转换后的图像不需要存储在磁盘上。在我们的实现中，转换后的图像是在 CPU 上用 Python 代码生成的，而 GPU 正在对上一批图像进行训练。因此，这些数据增强方案实际上在计算上是免费的（**现在来看，GPU 比 CPU 快很多很多，再在 CPU 上做数据增强，会非常耗时，成为性能瓶颈**）。

The first form of data augmentation

数据增强的第一种形式包括生成图像

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.

平移和水平反射。我们通过从256×256的图像中随机提取224×224的块（及其水平反射）并在这些提取的块上训练我们的网络来做到这一点。这将我们的训练集的大小增加了 2048 倍（**有 2048 种提取方法，但是提取出来的东西都长的差不多**），尽管由此产生的训练示例当然是高度相互依赖的。如果没有这个方案，我们的网络就会遭受严重的过拟合，这将迫使我们使用更小的网络。在测试时，网络通过提取 5 个224×224的块（四个角块和中心块）及其水平反射（因此总共有 10 个块），并对网络在十个块上的 softmax 层所做的预测求平均值。（**第一种方法从空间上去提取**）

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

第二种形式的数据增强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 训练集的 RGB 像素值集执行 PCA。对于每张训练图像，我们添加找到的主成分的倍数，其幅度与相应的特征值乘以从均值为零和标准差为 0.1 的高斯分布中提取的随机变量成正比。因此，对于每个 RGB 图像像素$I_{xy} = [I_{xy}^R, I_{xy}^G, I_{xy}^B]^T$，我们添加以下数量：

$$[P_1, P_2, P_3][\alpha_1\lambda_1, \alpha_2\lambda_2, \alpha_3\lambda_3]^T$$

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%.

其中P_i和λ_i分别是 RGB 像素值的3×3的协方差矩阵的特征向量和特征值，α_i是前面提到的随机变量。对于特定训练图像的所有像素，每个α_i仅绘制一次，直到再次使用该图像进行训练，然后重新绘制。该方案近似地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的变化是不变的。该方案将 top-1 错误率降低了 1%以上。（**第二种方法对颜色通道数做一些变化**）

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 backpropagation. 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.

结合许多不同模型的预测是减少测试错误的一种非常成功的方法[1, 3]，但对于已经需要几天时间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组合版本，在训练期间只需要大约两倍的成本。最近引入的称为“dropout”的技术[10]，包括对每个隐藏神经元以 0.5 的概率将输出设置为零。以这种方式“dropped out”的神经元不参与前向传播，也不参与反向传播。因此，每次表示输入时，神经网络都会对不同的架构进行采样，但所有这些架构共享权重（**作者觉得每次会得到不同的模型，等价于对多个不同的模型做融合，但后来研究发现，dropout 更多的不是做融合，而是作为一个正则项**）。这种技术减少了神经元复杂的协同适应，因为神经元不能依赖于特定其他神经元的存在。因此，它被迫学习与其他神经元的许多不同随机子集相结合使用的更强大的特征。在测试时，我们使用所有的神经元，但将它们的输出乘以 0.5，这是对多指数的 dropout 网络产生的预测分布的几何平均值的合理近似。

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 的前两个全连接层中使用了 dropout。如果没有 dropout，我们的网络就会出现严重的过拟合。Dropout 大约使收敛所需的迭代次数加倍（**dropout 并不是只有优点，需要付出一些代价**）。

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

我们使用 SGD 来训练我们的模型（**那个时候 SGD 因调参较复杂而被使用的较少，但后来研究发现，SGD 训练过程中的噪音是对模型的泛化能力有好处的，所以现在深度学习中 SGD 使用较多**），批量大小为 128 个样例，momentum 为 0.9，weight decay 为

merely a regularizer: it reduces the model's training error. The update rule for weight w was

0.0005。我们发现这种少量的权重衰减对于模型学习很重要。换句话说，这里的权重衰减不仅仅是一个正则化工具：它减少了模型的训练误差。权重 w 的更新规则是

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

$$\mathrm{w}_{i+1} := \mathrm{w}_i + v_{i+1}$$

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

其中i是迭代索引（**当前的迭代次数**），v 是 momentum 变量，$\in$ 是学习率，$\left< \frac{\partial L}{\partial \mathrm{w}} |_{w_i} \right>_{D_i}$ 是目标关于w的导数的第i个批次D_i的平均值，在w_i处评估。

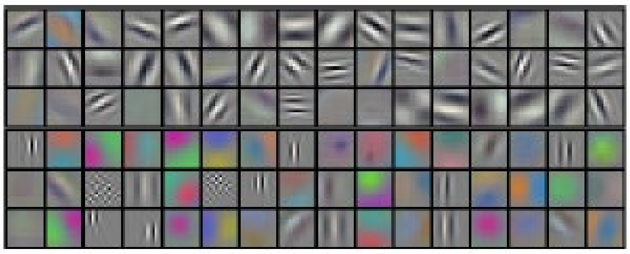


Figure 3: 96 convolutional kernels of size $11 \times 11 \times 3$ learned by the first convolutional layer on the $224 \times 224 \times 3$ input images. The top 48 kernels were learned on GPU 1 while the bottom 48 kernels were learned on GPU 2. See Section 6.1 for details.

图 3：第一个卷积层在$224 \times 224 \times 3$的输入图像上学习了 96 个大小为$11 \times 11 \times 3$的卷积核。前 48 个卷积核是在 GPU 1 上学习的，而后 48 个卷积核是在 GPU 2 上学习的。详见 6.1 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 的高斯分布初始化每一层的权重。我们将第二、第四和第五卷积层以及全连接隐藏层中的神经元偏差初始化为常数 1（**比较新奇，在这篇论文之后使用较少**）。此初始化通过为 ReLU 提供正输入来加速学习的早期阶段。我们将其余层中的神经元偏差初始化为常数 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

我们对所有层使用相同的学习率，并在整个训练过程中手动调整。我们遵循的启发式方法是，当验证集错误率在当前学习率下停止降低时，将学习率除以 10。学习率初始化为 0.01，并在终止前减少 3 次（**在论文之后的很长一段时间大家都是这样做的，但现在来说，学习率一般都会选择先线性升高，再平滑的（如 cos）下降**）。我们通过 120 万

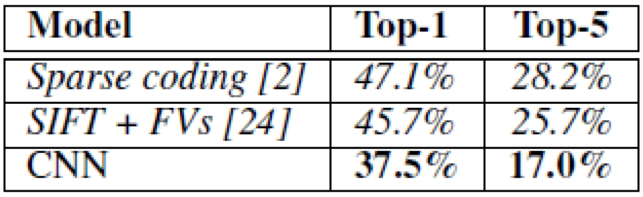
roughly 90 cycles through the training set of 1.2 million images, which took five to six days on two NVIDIA GTX 580 3GB GPUs.

张图像的训练集对网络进行了大约 90 轮的训练，这在两个 NVIDIA GTX 580 3GB GPU 上花费了五到六天的时间。

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].

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%和 17.0%的 top-1 和 top-5 测试集错误率 [5]。在 ILSVRC-2010 竞赛中的最佳性能分别为 47.1%和 28.2%，其方法是对在不同特征上训练的六个稀疏编码模型产生的预测求平均值[2]，在这之后发布的最佳结果分别为 45.7%和 25.7%，其方法是对基于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进行平均[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 测试集上的结果比较。*斜体*是其他人取得的最佳结果。

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

我们还使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 测试集的标签不是公开的，我们无法报告我们尝试过的所有模型的测试集错误率。在本段的其余部分，我们交替使用验证集和测试集错误率，因为根据我们的经验，它们的差异不超过 0.1%（见表 2）。本文介绍的 CNN 实现了 18.2%的 top-5 错误率。对五个类似的 CNN 的预测结果进行平均得到 16.4%的错误率。训练一个 CNN 在最后一个池化层上有一个额外的第六个卷积层，对整个 ImageNet 2011 年秋季版本（15M 张图像，22K 个类别）进行分类，然后在 ILSVRC-2012 上对其进行“微调”，错误率为 16.6%。将在整个 2011 年秋季发布的版本中预训练的两个 CNN 的预测与上述五个 CNN 的预测平均得出 15.3%的错误率。第二佳参赛作品的错误率为 26.2%，

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].

其方法是对在不同类型的密集采样特征计算的 FV 上训练的几个分类器的预测结果进行平均[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.

表 2：ILSVRC-2012 验证集和测试集上的错误率比较。*斜体*是其他人取得的最佳结果。带有星号*的模型经过"预训练"以对整个 ImageNet 2011 秋季版本进行分类。详见第 6 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，我们还报告了 2009 年秋季版 ImageNet 的错误率，该版本包含 10,184 个类别和 890 万张图像。在这个数据集上，我们遵循文献中的惯例，使用一半的图像进行训练，一半的图像进行测试。由于没有既定的测试集，我们的分割必然与之前作者使用的分割不同，但这不会显著影响结果。我们在这个数据集上的 top-1 和 top-5 错误率分别为 67.4%和 40.9%，通过上述网络获得，但在最后一个池化层上增加了第六个卷积层。在这个数据集上发布的最好结果是 78.1%和 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

图 3 显示了网络的两个数据连接层学习的卷积核。该网络已经学习了各种频率和方向选择性内核，以及各种彩色斑点。请注意两个 GPU 表现出的特殊化，这是第 3.5 节中描述的受限连接的结果。GPU 1 上的内核在很大程度上与颜色无关，而 GPU 2 上的内核在很大程度上与颜色相关。这种特殊化发生在每次运行期间，并且独立于任何特定

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).

的随机权重初始化（以 GPU 的重新编号为模）。（**这里比较神奇，后来也没有人解释，很玄学**）

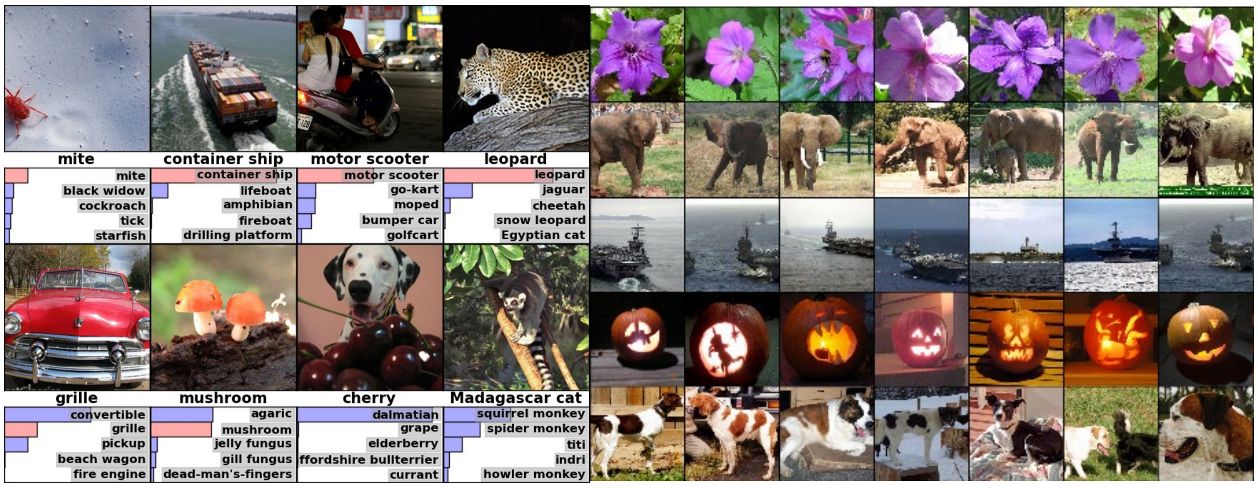


Figure 4: (**Left**) Eight ILSVRC-2010 test images and the five labels considered most probable by our model. The correct label is written under each image, and the probability assigned to the correct label is also shown with a red bar (if it happens to be in the top 5). (**Right**) Five ILSVRC-2010 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**）八张 ILSVRC-2010 测试图像以及我们的模型认为每张图像最有可能的五个标签。正确的标签写在每张图像下，分配给正确标签的概率也用红色条显示（如果它恰好在前 5 个）。(**右**）第一列中有五个 ILSVRC-2010 测试集图像。剩余的列显示了六个训练图像，它们在最后一个隐藏层中产生的特征向量，与测试图像的特征向量的欧几里得距离最小。

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 的左半边中，我们通过计算八张测试图像的前 5 名的预测结果来定性地评估网络所学到的东西。请注意即使是偏离中心的物体，例如左上角的螨虫，也可以被网络识别。大多数前 5 个标签看起来都合理。例如，只有其他类型的猫才被认为是豹子的合理标签。在某些情况下（grille、cherry），照片的预期焦点确实存在歧义。

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

探索网络视觉知识的另一种方法是考虑图像在最后一个 4096 维隐藏层引起的特征激活。如果两幅图像产生具有小的欧几里德差距的特征激活向量，我们可以说神经网络的更高级别认为它们是相似的。图 4 显示了来自测试集的五幅图像和来自训练集的

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.

六幅图像，根据该度量，它们与每幅图像最相似。请注意，在像素级别，检索到的训练图像在 L2 中通常与第一列中的查询图像不接近。例如，检索到的狗和大象以各种姿势出现。我们在补充材料中展示了更多测试图像的结果。

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 autoencoders 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.

通过使用两个 4096 维实值向量之间的欧几里德距离计算相似性是低效的，但可以通过训练自动编码器将这些向量压缩为短二进制代码来提高效率。这应该产生比将自动编码器应用于原始像素更好的图像检索方法[14]，它不使用图像标签，因此倾向于检索具有相似边缘模式的图像，无论它们在语义上是否相似。

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.

(只有讨论，没有结论，是个比较少见的写法)我们的结果表明，一个大型深度卷积神经网络能够仅使用监督学习在极具挑战性的数据集上取得破纪录的结果。值得注意的是，如果删除单个卷积层，我们的网络性能会下降。 例如，移除任何中间层都会导致网络的 top-1 性能下降约 2%。所以深度对于实现我们的结果真的很重要。（**宽度也很重要，要有较好的高宽比**）

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

为了简化我们的实验，我们没有使用任何无监督的预训练（**这句话让整个深度学习在非常长的一段时间内主要关注有标号的数据，即 Supervised Learning；如今 bert 等在 nlp 领域的兴起让大家的目光重新回到 Unsupervised Learning 上**），即使我们预计它会有所帮助，特别是如果我们获得足够的计算能力来显着增加网络的规模，而没有相应地增加标记数据量。到目前为止，我们的

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.

结果已经有所改善，因为我们已经扩大了我们的网络并对其进行了更长时间的训练，但为了匹配人类视觉系统的 infero-temporal 路径，我们还有许多数量级的工作要做（**现在回头看，在物体识别这个方面，深度学习已经做的比人类好多了**）。最终，我们希望在视频序列上使用非常大且深的卷积网络，其中时间结构（**时序信息**）提供非常有用的信息，而这些信息在静态图像中缺失或不那么明显（**video 数据就算放到现在也是一个比较难的工作**）。