第8章
深度学习

深度学习是加深了层的深度神经网络。基于之前介绍的网络，只需通过叠加层，就可以创建深度网络。本章我们将看一下深度学习的性质、课题和可能性，然后对当前的深度学习进行概括性的说明。

8.1 加深网络

关于神经网络，我们已经学了很多东西，比如构成神经网络的各种层、学习时的有效技巧、对图像特别有效的CNN、参数的最优化方法等，这些都是深度学习中的重要技术。本节我们将这些已经学过的技术汇总起来，创建一个深度网络，挑战MNIST数据集的手写数字识别。

8.1.1 向更深的网络出发

话不多说，这里我们来创建一个如图8-1所示的网络结构的CNN（一个比之前的网络都深的网络）。这个网络参考了下一节要介绍的VGG。

如图8-1所示，这个网络的层比之前实现的网络都更深。这里使用的卷积层全都是3×3的小型滤波器，特点是随着层的加深，通道数变大（卷积层的通道数从前面的层开始按顺序以16、16、32、32、64、64的方式增加）。此外，如图8-1所示，插入了池化层，以逐渐减小中间数据的空间大小；并且，后面的全连接层中使用了Dropout层。

图8-1 进行手写数字识别的深度CNN

这个网络使用He初始值作为权重的初始值，使用Adam更新权重参数。把上述内容总结起来，这个网络有如下特点。

- 基于3×3的小型滤波器的卷积层。
- 激活函数是ReLU。
- 全连接层的后面使用Dropout层。
- 基于Adam的最优化。
- 使用He初始值作为权重初始值。

从这些特征中可以看出，图8-1的网络中使用了多个之前介绍的神经网络技术。现在，我们使用这个网络进行学习。先说一下结论，这个网络的识别精度为99.38% ①，可以说是非常优秀的性能了！

① 最终的识别精度有少许偏差，不过在这个网络中，识别精度大体上都会超过99%。

实现图8–1的网络的源代码在ch08/deep_convnet.py中，训练用的代码在ch08/train_deepnet.py中。虽然使用这些代码可以重现这里进行的学习，不过深度网络的学习需要花费较多的时间（大概要半天以上）。本书以ch08/deep_conv_net_params.pkl的形式给出了学习完的权重参数。刚才的deep_convnet.py备有读入学习完的参数的功能，请根据需要进行使用。

图8-1的网络的错误识别率只有0.62%。这里我们实际看一下在什么样的图像上发生了识别错误。图8-2中显示了识别错误的例子。

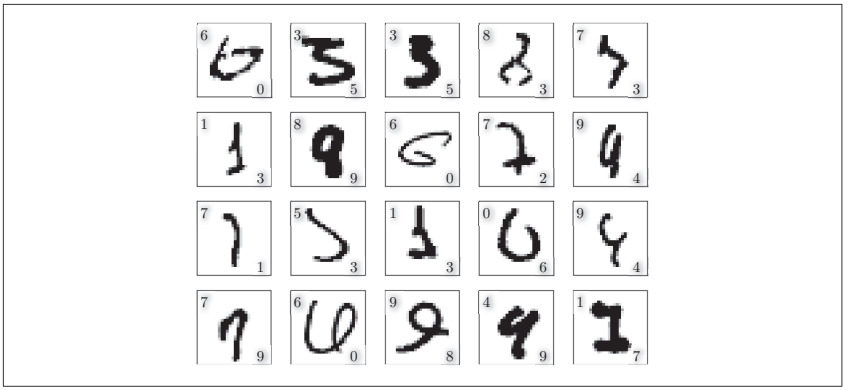

图8-2 识别错误的图像的例子：各个图像的左上角显示了正确解标签，右下角显示了本网络的推理结果

观察图8-2可知，这些图像对于我们人类而言也很难判断。实际上，这里面有几个图像很难判断是哪个数字，即使是我们人类，也同样会犯“识别错误”。比如，左上角的图像（正确解是“6”）看上去像“0”，它旁边的图像（正确解是“3”）看上去像“5”。整体上，“1”和“7”、“0”和“6”、“3”和“5”的组合比较容易混淆。通过这些例子，相信大家可以理解为何会发生识别错误了吧。

这次的深度CNN尽管识别精度很高，但是对于某些图像，也犯了和人类同样的“识别错误”。从这一点上，我们也可以感受到深度CNN中蕴藏着巨大的可能性。

8.1.2 进一步提高识别精度

在一个标题为“What is the class of this image ?”的网站[32]上，以排行榜的形式刊登了目前为止通过论文等渠道发表的针对各种数据集的方法的识别精度（图8-3）。

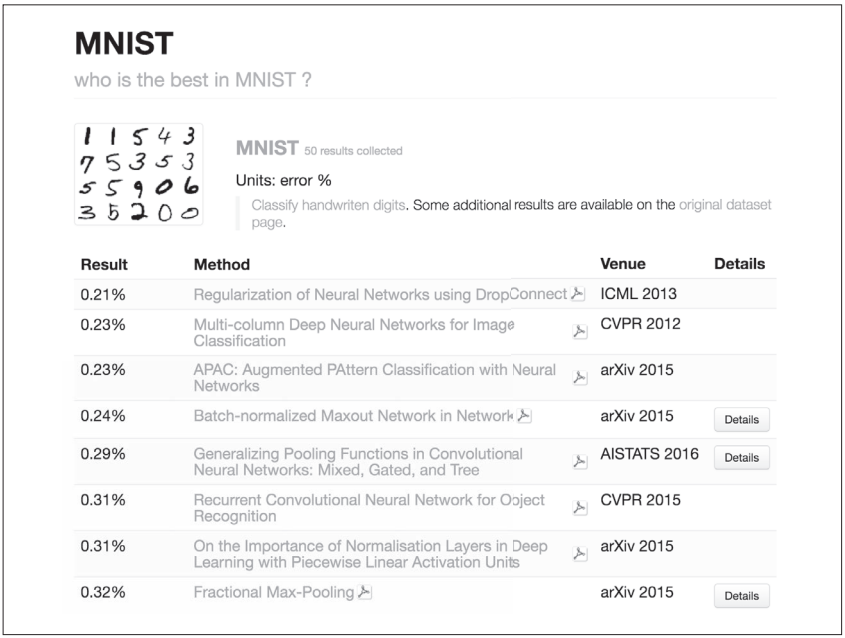

图8-3 针对MNIST数据集的各种方法的排行（引自文献[32]：2016年6月）

观察图8-3的排行结果，可以发现“Neural Networks”“Deep”“Convolutional”等关键词特别显眼。实际上，排行榜上的前几名大都是基于CNN的方法。顺便说一下，截止到2016年6月，对MNIST数据集的最高识别精度是99.79%（错误识别率为0.21%），该方法也是以CNN为基础的[33]。不过，它用的CNN并不是特别深层的网络（卷积层为2层、全连接层为2层的网络）。

对于MNIST数据集，层不用特别深就获得了（目前）最高的识别精度。一般认为，这是因为对于手写数字识别这样一个比较简单的任务，没有必要将网络的表现力提高到那么高的程度。因此，可以说加深层的好处并不大。而之后要介绍的大规模的一般物体识别的情况，因为问题复杂，所以加深层对提高识别精度大有裨益。

参考刚才排行榜中前几名的方法，可以发现进一步提高识别精度的技术和线索。比如，集成学习、学习率衰减、**Data Augmentation**（数据扩充）等都有助于提高识别精度。尤其是Data Augmentation，虽然方法很简单，但在提高识别精度上效果显著。

Data Augmentation基于算法“人为地”扩充输入图像（训练图像）。具体地说，如图8-4所示，对于输入图像，通过施加旋转、垂直或水平方向上的移动等微小变化，增加图像的数量。这在数据集的图像数量有限时尤其有效。

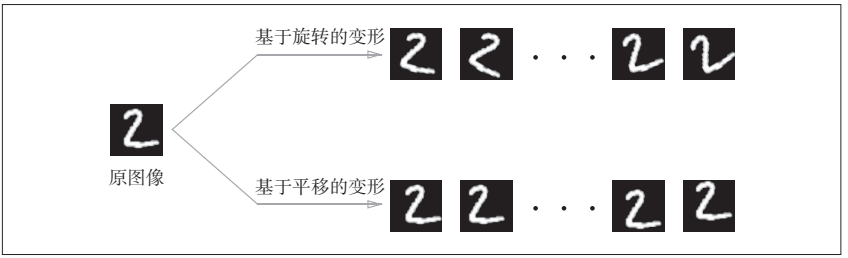

图8-4 Data Augmentation的例子

除了如图8-4所示的变形之外，Data Augmentation还可以通过其他各种方法扩充图像，比如裁剪图像的“crop处理”、将图像左右翻转的“flip处理”[①]等。对于一般的图像，施加亮度等外观上的变化、放大缩小等尺度上的变化也是有效的。不管怎样，通过Data Augmentation巧妙地增加训练图像，就可以提高深度学习的识别精度。虽然这个看上去只是一个简单的技巧，不过经常会有很好的效果。这里，我们不进行Data Augmentation的实现，不

① flip处理只在不需要考虑图像对称性的情况下有效。

过这个技巧的实现比较简单，有兴趣的读者请自己试一下。

8.1.3 加深层的动机

关于加深层的重要性，现状是理论研究还不够透彻。尽管目前相关理论还比较贫乏，但是有几点可以从过往的研究和实验中得以解释（虽然有一些直观）。本节就加深层的重要性，给出一些增补性的数据和说明。

首先，从以ILSVRC为代表的大规模图像识别的比赛结果中可以看出加深层的重要性（详细内容请参考下一节）。这种比赛的结果显示，最近前几名的方法多是基于深度学习的，并且有逐渐加深网络的层的趋势。也就是说，可以看到层越深，识别性能也越高。

下面我们说一下加深层的好处。其中一个好处就是可以减少网络的参数数量。说得详细一点，就是与没有加深层的网络相比，加深了层的网络可以用更少的参数达到同等水平（或者更强）的表现力。这一点结合卷积运算中的滤波器大小来思考就好理解了。比如，图8-5展示了由5 × 5的滤波器构成的卷积层。

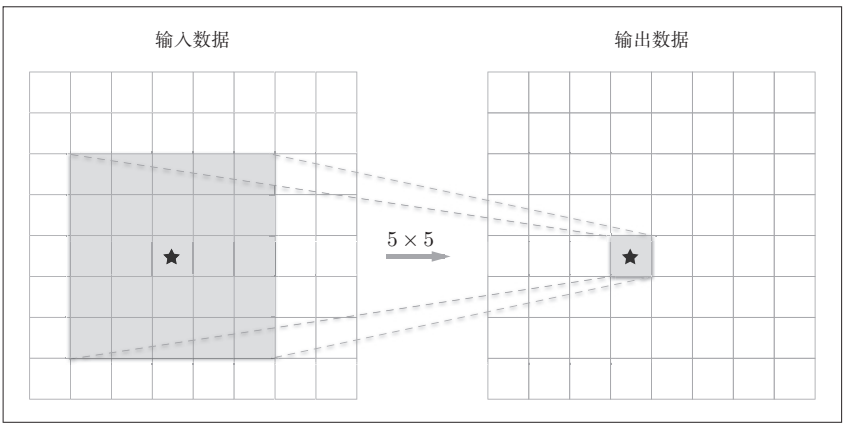

图8-5 5×5的卷积运算的例子

这里希望大家考虑一下输出数据的各个节点是从输入数据的哪个区域计算出来的。显然，在图8-5的例子中，每个输出节点都是从输入数据的某个5×5的区域算出来的。接下来我们思考一下图8-6中重复两次3×3的卷积运算的情形。此时，每个输出节点将由中间数据的某个3×3的区域计算出来。那么，中间数据的3×3的区域又是由前一个输入数据的哪个区域计算出来的呢？仔细观察图8-6，可知它对应一个5×5的区域。也就是说，图8-6的输出数据是“观察”了输入数据的某个5×5的区域后计算出来的。

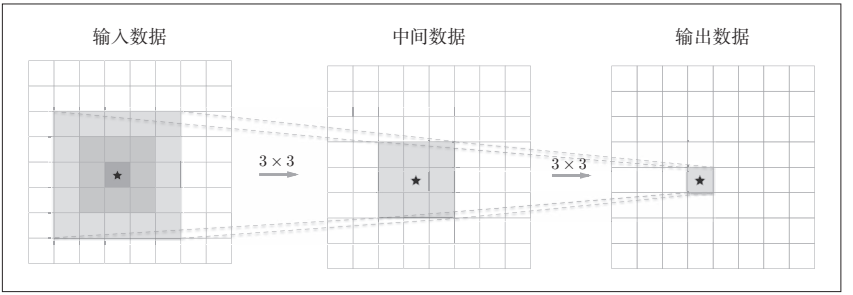

图8-6 重复两次3×3的卷积层的例子

一次5×5的卷积运算的区域可以由两次3×3的卷积运算抵充。并且，相对于前者的参数数量25（5×5），后者一共是18（$2 \times 3 \times 3$），通过叠加卷积层，参数数量减少了。而且，这个参数数量之差会随着层的加深而变大。比如，重复三次3×3的卷积运算时，参数的数量总共是27。而为了用一次卷积运算“观察”与之相同的区域，需要一个7×7的滤波器，此时的参数数量是49。

叠加小型滤波器来加深网络的好处是可以减少参数的数量，扩大**感受野**（receptive field，给神经元施加变化的某个局部空间区域）。并且，通过叠加层，将ReLU等激活函数夹在卷积层的中间，进一步提高了网络的表现力。这是因为向网络添加了基于激活函数的“非线性”表现力，通过非线性函数的叠加，可以表现更加复杂的东西。

加深层的另一个好处就是使学习更加高效。与没有加深层的网络相比，通过加深层，可以减少学习数据，从而高效地进行学习。为了直观地理解这一点，大家可以回忆一下7.6节的内容。7.6节中介绍了CNN的卷积层会分层次地提取信息。具体地说，在前面的卷积层中，神经元会对边缘等简单的形状有响应，随着层的加深，开始对纹理、物体部件等更加复杂的东西有响应。

我们先牢记这个网络的分层结构，然后考虑一下“狗”的识别问题。要用浅层网络解决这个问题的话，卷积层需要一下子理解很多“狗”的特征。“狗”有各种各样的种类，根据拍摄环境的不同，外观变化也很大。因此，要理解“狗”的特征，需要大量富有差异性的学习数据，而这会导致学习需要花费很多时间。

不过，通过加深网络，就可以分层次地分解需要学习的问题。因此，各层需要学习的问题就变成了更简单的问题。比如，最开始的层只要专注于学习边缘就好，这样一来，只需用较少的学习数据就可以高效地进行学习。这是为什么呢？因为和印有“狗”的照片相比，包含边缘的图像数量众多，并且边缘的模式比“狗”的模式结构更简单。

通过加深层，可以分层次地传递信息，这一点也很重要。比如，因为提取了边缘的层的下一层能够使用边缘的信息，所以应该能够高效地学习更加高级的模式。也就是说，通过加深层，可以将各层要学习的问题分解成容易解决的简单问题，从而可以进行高效的学习。

以上就是对加深层的重要性的增补性说明。不过，这里需要注意的是，近几年的深层化是由大数据、计算能力等即便加深层也能正确地进行学习的新技术和环境支撑的。

8.2 深度学习的小历史

一般认为，现在深度学习之所以受到大量关注，其契机是2012年举办的大规模图像识别大赛ILSVRC（ImageNet Large Scale Visual Recognition Challenge）。在那年的比赛中，基于深度学习的方法（通称AlexNet）以压倒性的优势胜出，彻底颠覆了以往的图像识别方法。2012年深度学习的这场逆

袭成为一个转折点，在之后的比赛中，深度学习一直活跃在舞台中央。本节我们以ILSVRC这个大规模图像识别比赛为轴，看一下深度学习最近的发展趋势。

8.2.1 ImageNet

ImageNet[25]是拥有超过100万张图像的数据集。如图8-7所示，它包含了各种各样的图像，并且每张图像都被关联了标签（类别名）。每年都会举办使用这个巨大数据集的ILSVRC图像识别大赛。

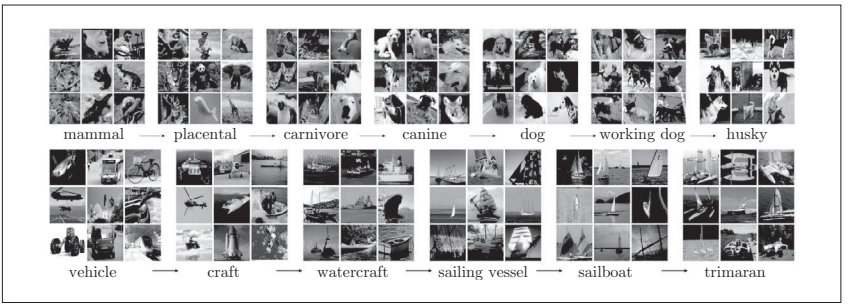

图8-7 大规模数据集ImageNet的数据例（引用自文献[25]）

ILSVRC大赛有多个测试项目，其中之一是“类别分类”（classification），在该项目中，会进行1000个类别的分类，比试识别精度。我们来看一下最近几年的ILSVRC大赛的类别分类项目的结果。图8-8中展示了从2010年到2015年的优胜队伍的成绩。这里，将前5类中出现正确解的情况视为“正确”，此时的错误识别率用柱形图来表示。

图8-8中需要注意的是，以2012年为界，之后基于深度学习的方法一直居于首位。实际上，我们发现2012年的AlexNet大幅降低了错误识别率。并且，此后基于深度学习的方法不断在提升识别精度。特别是2015年的ResNet（一个超过150层的深度网络）将错误识别率降低到了3.5%。据说这个结果甚至超过了普通人的识别能力。

这些年深度学习取得了不斐的成绩，其中VGG、GoogLeNet、ResNet

已广为人知，在与深度学习有关的各种场合都会遇到这些网络。下面我们就来简单地介绍一下这3个有名的网络。

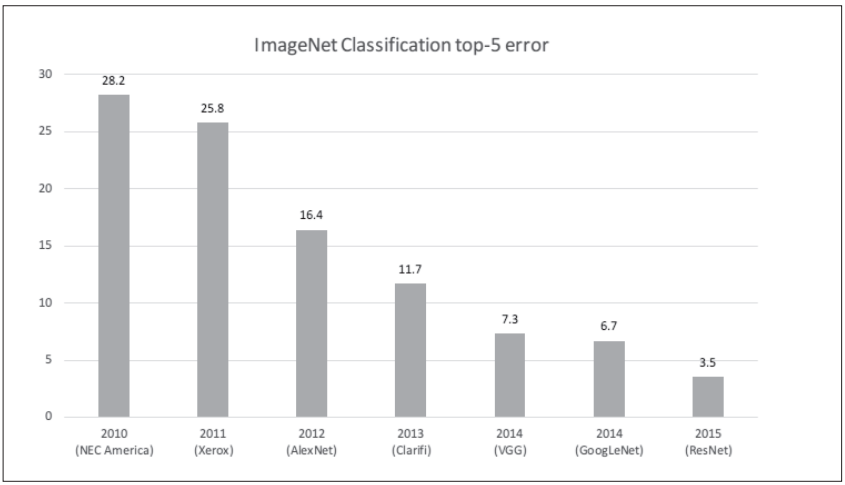

图8-8 ILSCRV优胜队伍的成绩演变：竖轴是错误识别率，横轴是年份。横轴的括号内是队伍名或者方法名

8.2.2 VGG

VGG是由卷积层和池化层构成的基础的CNN。不过，如图8-9所示，它的特点在于将有权重的层（卷积层或者全连接层）叠加至16层（或者19层），具备了深度（根据层的深度，有时也称为“VGG16”或“VGG19”）。

VGG中需要注意的地方是，基于3×3的小型滤波器的卷积层的运算是连续进行的。如图8-9所示，重复进行“卷积层重叠2次到4次，再通过池化层将大小减半”的处理，最后经由全连接层输出结果。

VGG在2014年的比赛中最终获得了第2名的成绩（下一节介绍的GoogleNet是2014年的第1名）。虽然在性能上不及GoogleNet，但因为VGG结构简单，应用性强，所以很多技术人员都喜欢使用基于VGG的网络。

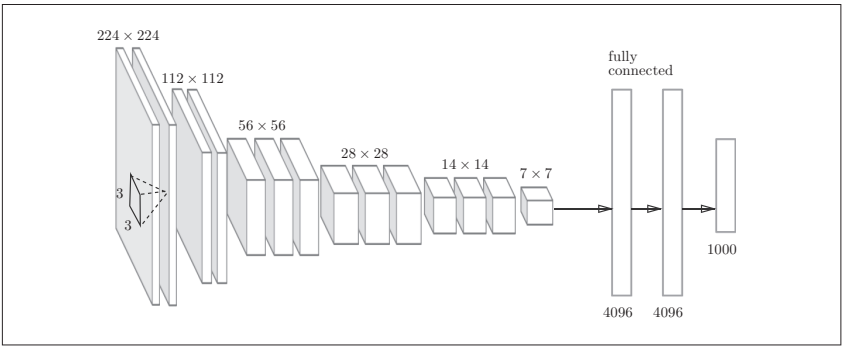

图8-9 VGG（根据文献[22]生成）

8.2.3 GoogLeNet

GoogLeNet的网络结构如图8-10所示。图中的矩形表示卷积层、池化层等。

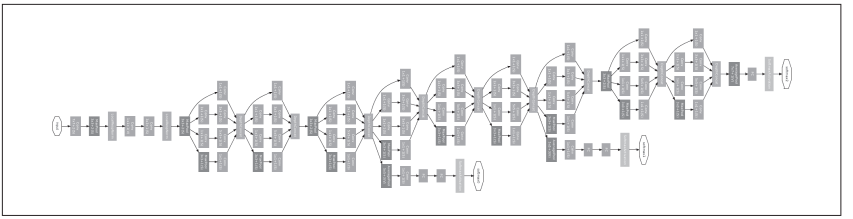

图8-10 GoogLeNet（引用自文献[23]）

只看图的话，这似乎是一个看上去非常复杂的网络结构，但实际上它基本上和之前介绍的CNN结构相同。不过，GoogLeNet的特征是，网络不仅在纵向上有深度，在横向上也有深度（广度）。

GoogLeNet在横向上有“宽度”，这称为“Inception结构”，以图8-11所示的结构为基础。

如图8-11所示，Inception结构使用了多个大小不同的滤波器（和池化），最后再合并它们的结果。GoogLeNet的特征就是将这个Inception结构用作一个构件（构成元素）。此外，在GoogLeNet中，很多地方都使用了大小为

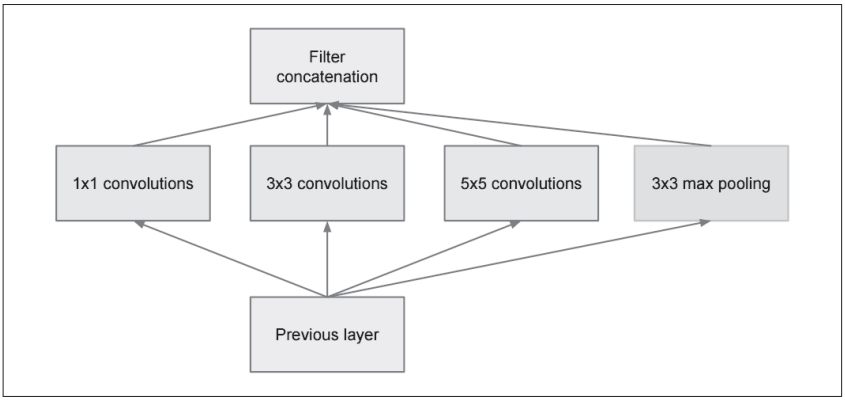

图8-11 GoogLeNet的Inception结构(引用自文献[23])

1×1的滤波器的卷积层。这个1×1的卷积运算通过在通道方向上减小大小，有助于减少参数和实现高速化处理(具体请参考原始论文[23])。

8.2.4 ResNet

ResNet[24]是微软团队开发的网络。它的特征在于具有比以前的网络更深的结构。

我们已经知道加深层对于提升性能很重要。但是，在深度学习中，过度加深层的话，很多情况下学习将不能顺利进行，导致最终性能不佳。ResNet中，为了解决这类问题，导入了“快捷结构”(也称为“捷径”或“小路”)。导入这个快捷结构后，就可以随着层的加深而不断提高性能了(当然，层的加深也是有限度的)。

如图8-12所示，快捷结构横跨(跳过)了输入数据的卷积层，将输入x合计到输出。

图8-12中，在连续2层的卷积层中，将输入x跳着连接至2层后的输出。这里的重点是，通过快捷结构，原来的2层卷积层的输出$\mathcal{F}(x)$变成了$\mathcal{F}(x)+x$。通过引入这种快捷结构，即使加深层，也能高效地学习。这是因为，通过快捷结构，反向传播时信号可以无衰减地传递。

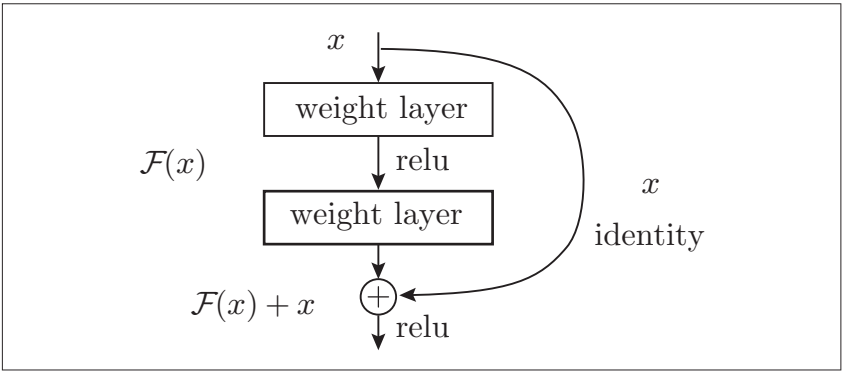

图8-12 ResNet的构成要素(引用自文献[24]):这里的“weight layer”是指卷积层

因为快捷结构只是原封不动地传递输入数据,所以反向传播时会将来自上游的梯度原封不动地传向下游。这里的重点是不对来自上游的梯度进行任何处理,将其原封不动地传向下游。因此,基于快捷结构,不用担心梯度会变小(或变大),能够向前一层传递“有意义的梯度”。通过这个快捷结构,之前因为加深层而导致的梯度变小的梯度消失问题就有望得到缓解。

ResNet以前面介绍过的VGG网络为基础,引入快捷结构以加深层,其结果如图8-13所示。

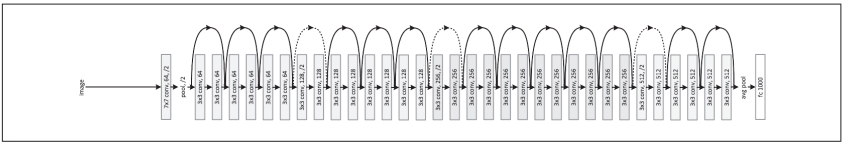

图8-13 ResNet(引用自文献[24]):方块对应3×3的卷积层,其特征在于引入了横跨层的快捷结构

如图8-13所示,ResNet通过以2个卷积层为间隔跳跃式地连接来加深层。另外,根据实验的结果,即便加深到150层以上,识别精度也会持续提高。并且,在ILSVRC大赛中,ResNet的错误识别率为3.5%(前5类中包含正确解这一精度下的错误识别率),令人称奇。

实践中经常会灵活应用使用ImageNet这个巨大的数据集学习到的权重数据，这称为**迁移学习**，将学习完的权重（的一部分）复制到其他神经网络，进行再学习（fine tuning）。比如，准备一个和VGG相同结构的网络，把学习完的权重作为初始值，以新数据集为对象，进行再学习。迁移学习在手头数据集较少时非常有效。

8.3 深度学习的高速化

随着大数据和网络的大规模化，深度学习需要进行大量的运算。虽然到目前为止，我们都是使用CPU进行计算的，但现实是只用CPU来应对深度学习无法令人放心。实际上，环视一下周围，大多数深度学习的框架都支持**GPU**（Graphics Processing Unit），可以高速地处理大量的运算。另外，最近的框架也开始支持多个GPU或多台机器上的分布式学习。本节我们将焦点放在深度学习的计算的高速化上，然后逐步展开。深度学习的实现在8.1节就结束了，本节要讨论的高速化（支持GPU等）并不进行实现。

8.3.1 需要努力解决的问题

在介绍深度学习的高速化之前，我们先来看一下深度学习中什么样的处理比较耗时。图8-14中以AlexNet的`forward`处理为对象，用饼图展示了各层所耗费的时间。

从图中可知，AlexNex中，大多数时间都被耗费在卷积层上。实际上，卷积层的处理时间加起来占GPU整体的95%，占CPU整体的89%！因此，如何高速、高效地进行卷积层中的运算是深度学习的一大课题。虽然图8-14是推理时的结果，不过学习时也一样，卷积层中会耗费大量时间。

正如7.2节介绍的那样，卷积层中进行的运算可以追溯至乘积累加运算。因此，深度学习的高速化的主要课题就变成了如何高速、高效地进行大量的乘积累加运算。

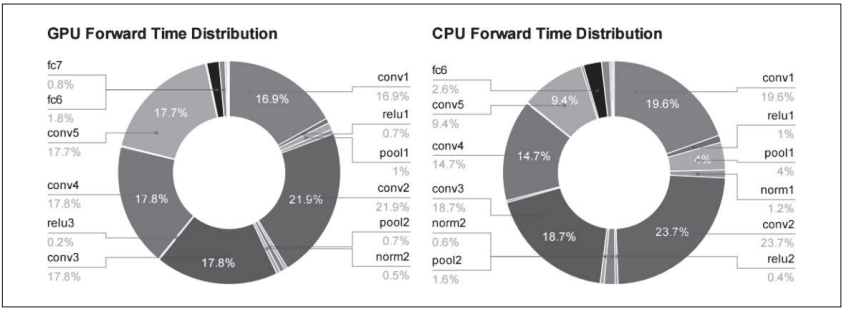

图8-14 AlexNet的forward处理中各层的时间比：左边是使用GPU的情况，右边是使用CPU的情况。图中的“conv”对应卷积层，“pool”对应池化层，“fc”对应全连接层，“norm”对应正规化层（引用自文献[26]）

8.3.2 基于GPU的高速化

GPU原本是作为图像专用的显卡使用的，但最近不仅用于图像处理，也用于通用的数值计算。由于GPU可以高速地进行并行数值计算，因此**GPU计算**的目标就是将这种压倒性的计算能力用于各种用途。所谓GPU计算，是指基于GPU进行通用的数值计算的操作。

深度学习中需要进行大量的乘积累加运算（或者大型矩阵的乘积运算）。这种大量的并行运算正是GPU所擅长的（反过来说，CPU比较擅长连续的、复杂的计算）。因此，与使用单个CPU相比，使用GPU进行深度学习的运算可以达到惊人的高速化。下面我们就来看一下基于GPU可以实现多大程度的高速化。图8-15是基于CPU和GPU进行AlexNet的学习时分别所需的时间。

从图中可知，使用CPU要花40天以上的时间，而使用GPU则可以将时间缩短至6天。此外，还可以看出，通过使用cuDNN这个最优化的库，可以进一步实现高速化。

GPU主要由NVIDIA和AMD两家公司提供。虽然两家的GPU都可以用于通用的数值计算，但与深度学习比较“亲近”的是NVIDIA的GPU。实际上，大多数深度学习框架只受益于NVIDIA的GPU。这是因为深度学习的框架中使用了NVIDIA提供的CUDA这个面向GPU计算的综合开发环境。

图8-15中出现的cuDNN是在CUDA上运行的库，它里面实现了为深度学习最优化过的函数等。

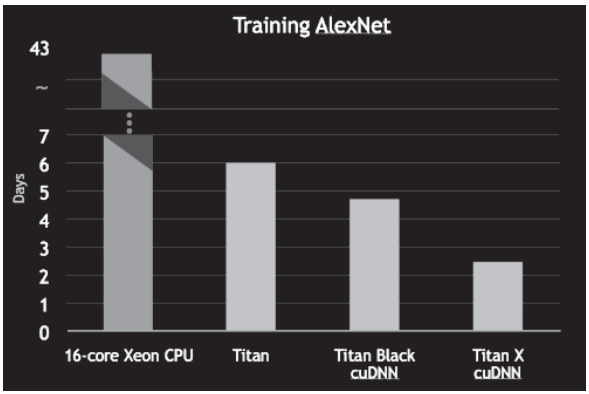

图8-15 使用CPU的“16-core Xeon CPU”和GPU的“Titan系列”进行AlexNet的学习时分别所需的时间（引用自文献[27]）

通过im2col可以将卷积层进行的运算转换为大型矩阵的乘积。这个im2col方式的实现对GPU来说是非常方便的实现方式。这是因为，相比按小规模的单位进行计算，GPU更擅长计算大规模的汇总好的数据。也就是说，通过基于im2col以大型矩阵的乘积的方式汇总计算，更容易发挥出GPU的能力。

8.3.3 分布式学习

虽然通过GPU可以实现深度学习运算的高速化，但即便如此，当网络较深时，学习还是需要几天到几周的时间。并且，前面也说过，深度学习伴随着很多试错。为了创建良好的网络，需要反复进行各种尝试，这样一来就必然会产生尽可能地缩短一次学习所需的时间的要求。于是，将深度学习的学习过程扩展开来的想法（也就是分布式学习）就变得重要起来。

为了进一步提高深度学习所需的计算的速度，可以考虑在多个GPU或

者多台机器上进行分布式计算。现在的深度学习框架中，出现了好几个支持多GPU或者多机器的分布式学习的框架。其中，Google的TensorFlow、微软的CNTK（Computational Network Toolki）在开发过程中高度重视分布式学习。以大型数据中心的低延迟・高吞吐网络作为支撑，基于这些框架的分布式学习呈现出惊人的效果。

基于分布式学习，可以达到何种程度的高速化呢？图8-16中显示了基于TensorFlow的分布式学习的效果。

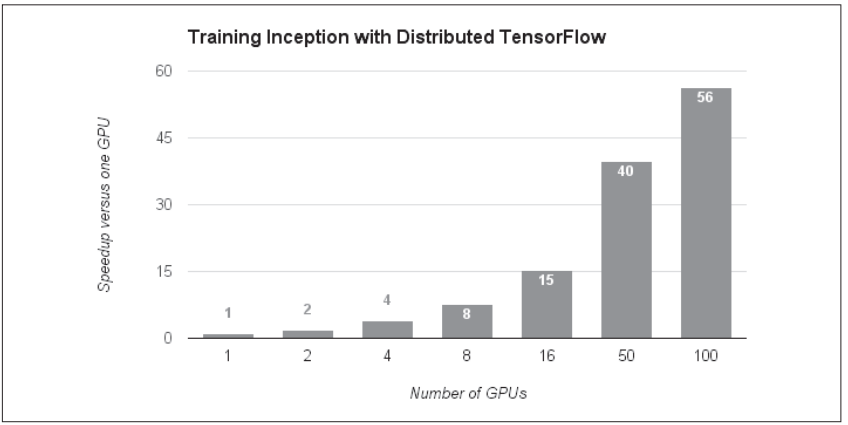

图8-16 基于TensorFlow的分布式学习的效果：横轴是GPU的个数，纵轴是与单个GPU相比时的高速化率（引用自文献[28]）

如图8-16所示，随着GPU个数的增加，学习速度也在提高。实际上，与使用1个GPU时相比，使用100个GPU（设置在多台机器上，共100个）似乎可以实现56倍的高速化！这意味着之前花费7天的学习只要3个小时就能完成，充分说明了分布式学习惊人的效果。

关于分布式学习，“如何进行分布式计算”是一个非常难的课题。它包含了机器间的通信、数据的同步等多个无法轻易解决的问题。可以将这些难题都交给TensorFlow等优秀的框架。这里，我们不讨论分布式学习的细节。关于分布式学习的技术性内容，请参考TensorFlow的技术论文（白皮书）等。

8.3.4 运算精度的位数缩减

在深度学习的高速化中，除了计算量之外，内存容量、总线带宽等也有可能成为瓶颈。关于内存容量，需要考虑将大量的权重参数或中间数据放在内存中。关于总线带宽，当流经GPU（或者CPU）总线的数据超过某个限制时，就会成为瓶颈。考虑到这些情况，我们希望尽可能减少流经网络的数据的位数。

计算机中为了表示实数，主要使用64位或者32位的浮点数。通过使用较多的位来表示数字，虽然数值计算时的误差造成的影响变小了，但计算的处理成本、内存使用量却相应地增加了，还给总线带宽带来了负荷。

关于数值精度（用几位数据表示数值），我们已经知道深度学习并不那么需要数值精度的位数。这是神经网络的一个重要性质。这个性质是基于神经网络的健壮性而产生的。这里所说的健壮性是指，比如，即便输入图像附有一些小的噪声，输出结果也仍然保持不变。可以认为，正是因为有了这个健壮性，流经网络的数据即便有所“劣化”，对输出结果的影响也较小。

计算机中表示小数时，有32位的单精度浮点数和64位的双精度浮点数等格式。根据以往的实验结果，在深度学习中，即便是16位的**半精度浮点数**（half float），也可以顺利地进行学习[30]。实际上，NVIDIA的下一代GPU框架Pascal也支持半精度浮点数的运算，由此可以认为今后半精度浮点数将被作为标准使用。

NVIDIA的Maxwell GPU虽然支持半精度浮点数的存储（保存数据的功能），但是运算本身不是用16位进行的。下一代的Pascal框架，因为运算也是用16位进行的，所以只用半精度浮点数进行计算，就有望实现超过上一代GPU约2倍的高速化。

以往的深度学习的实现中并没有注意数值的精度，不过Python中一般使用64位的浮点数。NumPy中提供了16位的半精度浮点数类型（不过，只有16位类型的存储，运算本身不用16位进行），即便使用NumPy的半精度浮点数，识别精度也不会下降。相关的论证也很简单，有兴趣的读者请参考`ch08/half_float_network.py`。

关于深度学习的位数缩减，到目前为止已有若干研究。最近有人提出了用1位来表示权重和中间数据的Binarized Neural Networks方法[31]。为了实现深度学习的高速化，位数缩减是今后必须关注的一个课题，特别是在面向嵌入式应用程序中使用深度学习时，位数缩减非常重要。

8.4 深度学习的应用案例

前面，作为使用深度学习的例子，我们主要讨论了手写数字识别的图像类别分类问题（称为“物体识别”）。不过，深度学习并不局限于物体识别，还可以应用于各种各样的问题。此外，在图像、语音、自然语言等各个不同的领域，深度学习都展现了优异的性能。本节将以计算机视觉这个领域为中心，介绍几个深度学习能做的事情（应用）。

8.4.1 物体检测

物体检测是从图像中确定物体的位置，并进行分类的问题。如图8-17所示，要从图像中确定物体的种类和物体的位置。

图8-17 物体检测的例子（引用自文献[34]）

观察图8-17可知，物体检测是比物体识别更难的问题。之前介绍的物体识别是以整个图像为对象的，但是物体检测需要从图像中确定类别的位置，而且还有可能存在多个物体。

对于这样的物体检测问题，人们提出了多个基于CNN的方法。这些方法展示了非常优异的性能，并且证明了在物体检测的问题上，深度学习是非常有效的。

在使用CNN进行物体检测的方法中，有一个叫作R-CNN[35]的有名的方法。图8-18显示了R-CNN的处理流。

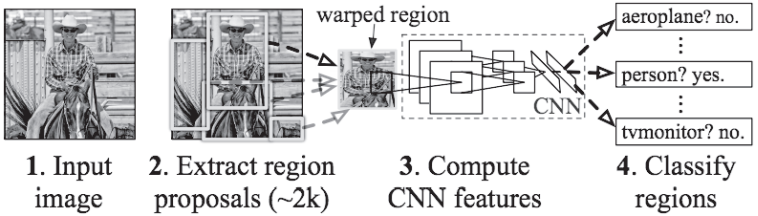

图8-18 R-CNN的处理流（引用自文献[35]）

希望大家注意图中的“2.Extract region proposals”（候选区域的提取）和“3.Compute CNN features”（CNN特征的计算）的处理部分。这里，首先（以某种方法）找出形似物体的区域，然后对提取出的区域应用CNN进行分类。R-CNN中会将图像变形为正方形，或者在分类时使用SVM（支持向量机），实际的处理流会稍微复杂一些，不过从宏观上看，也是由刚才的两个处理（候选区域的提取和CNN特征的计算）构成的。

在R-CNN的前半部分的处理——候选区域的提取（发现形似目标物体的处理）中，可以使用计算机视觉领域积累的各种各样的方法。R-CNN的论文中使用了一种被称为Selective Search的方法，最近还提出了一种基于CNN来进行候选区域提取的Faster R-CNN方法[36]。Faster R-CNN用一个CNN来完成所有处理，使得高速处理成为可能。

8.4.2 图像分割

图像分割是指在像素水平上对图像进行分类。如图8-19所示，使用以像素为单位对各个对象分别着色的监督数据进行学习。然后，在推理时，对输入图像的所有像素进行分类。

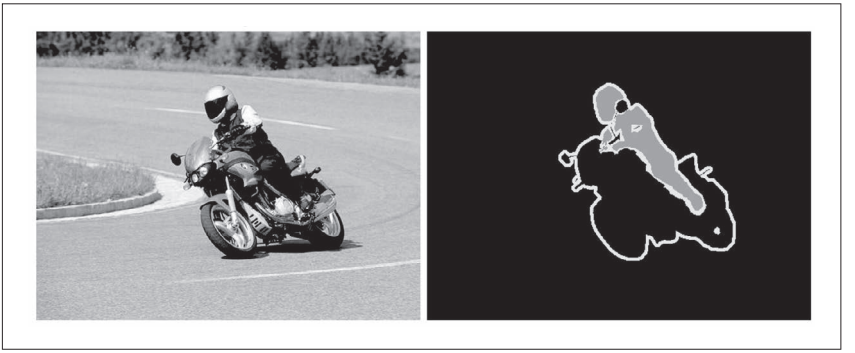

图8-19 图像分割的例子（引用自文献[34]）：左边是输入图像，右边是监督用的带标签图像

之前实现的神经网络是对图像整体进行了分类，要将它落实到像素水平的话，该怎么做呢？

要基于神经网络进行图像分割，最简单的方法是以所有像素为对象，对每个像素执行推理处理。比如，准备一个对某个矩形区域中心的像素进行分类的网络，以所有像素为对象执行推理处理。正如大家能想到的，这样的方法需要按照像素数量进行相应次`forward`处理，因而需要耗费大量的时间（正确地说，卷积运算中会发生重复计算很多区域的无意义的计算）。为了解决这个无意义的计算问题，有人提出了一个名为FCN（Fully Convolutional Network）[37]的方法。该方法通过一次`forward`处理，对所有像素进行分类（图8-20）。

FCN的字面意思是“全部由卷积层构成的网络”。相对于一般的CNN包含全连接层，FCN将全连接层替换成发挥相同作用的卷积层。在物体识别中使用的网络的全连接层中，中间数据的空间容量被作为排成一列的节点进

行处理，而只由卷积层构成的网络中，空间容量可以保持原样直到最后的输出。

如图8-20所示，FCN的特征在于最后导入了扩大空间大小的处理。基于这个处理，变小了的中间数据可以一下子扩大到和输入图像一样的大小。FCN最后进行的扩大处理是基于双线性插值法的扩大（双线性插值扩大）。FCN中，这个双线性插值扩大是通过去卷积（逆卷积运算）来实现的（细节请参考FCN的论文[37]）。

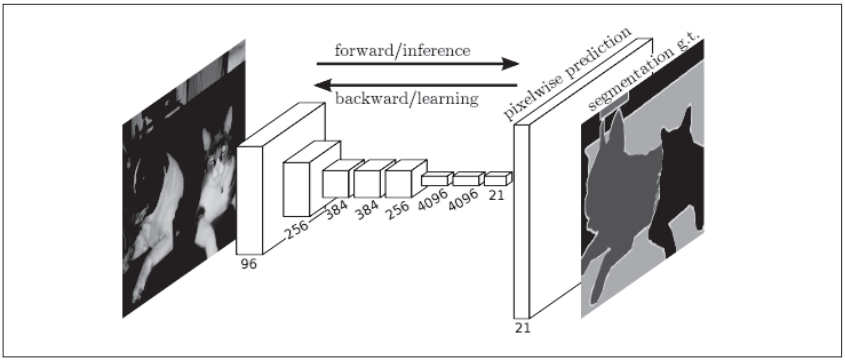

图8-20 FCN的概略图（引用自文献[37]）

全连接层中，输出和全部的输入相连。使用卷积层也可以实现与此结构完全相同的连接。比如，针对输入大小是32×10×10（通道数32、高10、长10）的数据的全连接层可以替换成滤波器大小为32×10×10的卷积层。如果全连接层的输出节点数是100，那么在卷积层准备100个32×10×10的滤波器就可以实现完全相同的处理。像这样，全连接层可以替换成进行相同处理的卷积层。

8.4.3 图像标题的生成

有一项融合了计算机视觉和自然语言的有趣的研究，该研究如图8-21所示，给出一个图像后，会自动生成介绍这个图像的文字（图像的标题）。

给出一个图像后，会像图8-21一样自动生成表示该图像内容的文本。比如，左上角的第一幅图像生成了文本“A person riding a motorcycle on a

图8-21 基于深度学习的图像标题生成的例子(引用自文献[38])

dirt road."(在没有铺装的道路上骑摩托车的人),而且这个文本只从该图像自动生成。文本的内容和图像确实是一致的。并且,令人惊讶的是,除了"骑摩托车"之外,连"没有铺装的道路"都被正确理解了。

一个基于深度学习生成图像标题的代表性方法是被称为NIC(Neural Image Caption)的模型。如图8-22所示,NIC由深层的CNN和处理自然语言的RNN(Recurrent Neural Network)构成。RNN是呈递归式连接的网络,经常被用于自然语言、时间序列数据等连续性的数据上。

NIC基于CNN从图像中提取特征,并将这个特征传给RNN。RNN以CNN提取出的特征为初始值,递归地生成文本。这里,我们不深入讨论技术上的细节,不过基本上NIC是组合了两个神经网络(CNN和RNN)的简单结构。基于NIC,可以生成惊人的高精度的图像标题。我们将组合图像和自然语言等多种信息进行的处理称为**多模态处理**。多模态处理是近年来备受关注的一个领域。

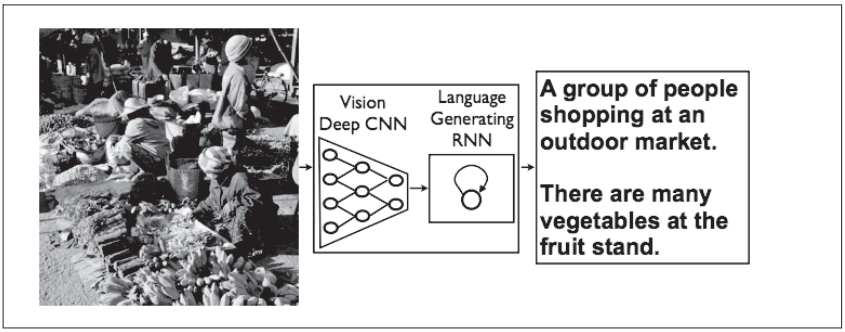

图8-22 Neural Image Caption(NIC)的整体结构(引用自文献[38])

RNN的R表示Recurrent(递归的)。这个递归指的是神经网络的递归的网络结构。根据这个递归结构，神经网络会受到之前生成的信息的影响(换句话说，会记忆过去的信息)，这是RNN的特征。比如，生成“我”这个词之后，下一个要生成的词受到“我”这个词的影响，生成了“要”；然后，再受到前面生成的“我要”的影响，生成了“睡觉”这个词。对于自然语言、时间序列数据等连续性的数据，RNN以记忆过去的信息的方式运行。

8.5 深度学习的未来

深度学习已经不再局限于以往的领域，开始逐渐应用于各个领域。本节将介绍几个揭示了深度学习的可能性和未来的研究。

8.5.1 图像风格变换

有一项研究是使用深度学习来“绘制”带有艺术气息的画。如图8-23所示，输入两个图像后，会生成一个新的图像。两个输入图像中，一个称为“内容图像”，另一个称为“风格图像”。

如图8-23所示，如果指定将梵高的绘画风格应用于内容图像，深度学习就会按照指示绘制出新的画作。此项研究出自论文“A Neural Algorithm of

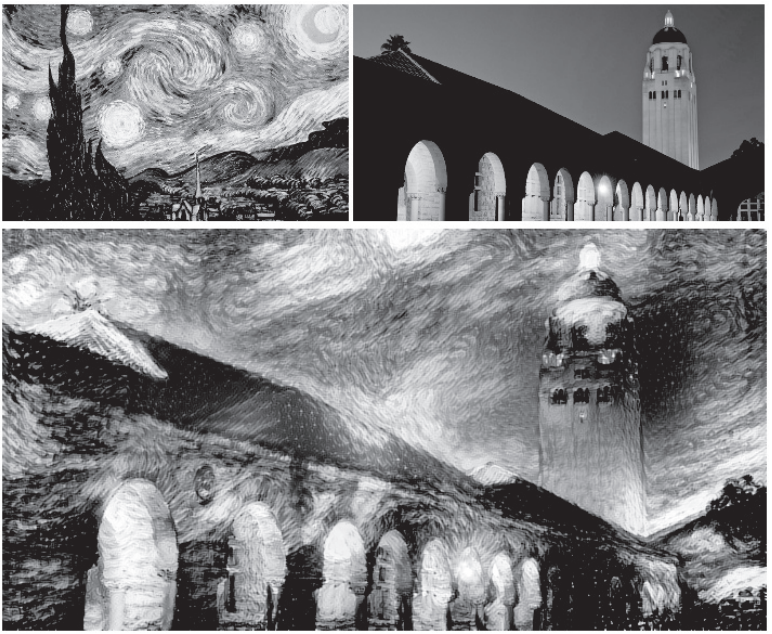

图8-23 基于论文“A Neural Algorithm of Artistic Style”的图像风格变换的例子：左上角是风格图像，右上角是内容图像，下面的图像是新生成的图像（图像引用自文献[40]）

Artistic Style”[39]，一经发表就受到全世界的广泛关注。

这里我们不会介绍这项研究的详细内容，只是叙述一下这个技术的大致框架，即刚才的方法是在学习过程中使网络的中间数据近似内容图像的中间数据。这样一来，就可以使输入图像近似内容图像的形状。此外，为了从风格图像中吸收风格，导入了风格矩阵的概念。通过在学习过程中减小风格矩阵的偏差，就可以使输入图像接近梵高的风格。

8.5.2 图像的生成

刚才的图像风格变换的例子在生成新的图像时输入了两个图像。不同于这种研究，现在有一种研究是生成新的图像时不需要任何图像（虽然需要事

先使用大量的图像进行学习，但在“画”新图像时不需要任何图像）。比如，基于深度学习，可以实现从零生成“卧室”的图像。图8-24中展示的图像是基于**DCGAN**（Deep Convolutional Generative Adversarial Network）[41]方法生成的卧室图像的例子。

图8-24 基于DCGAN生成的新的卧室图像（引用自文献[41]）

图8-24的图像可能看上去像是真的照片，但其实这些图像都是基于DCGAN新生成的图像。也就是说，DCGAN生成的图像是谁都没有见过的图像（学习数据中没有的图像），是从零生成的新图像。

能画出以假乱真的图像的DCGAN会将图像的生成过程模型化。使用大量图像（比如，印有卧室的大量图像）训练这个模型，学习结束后，使用这个模型，就可以生成新的图像。

DCGAN中使用了深度学习，其技术要点是使用了Generator（生成者）和Discriminator（识别者）这两个神经网络。Generator生成近似真品的图像，Discriminator判别它是不是真图像（是Generator生成的图像还是实际拍摄的图像）。像这样，通过让两者以竞争的方式学习，Generator会学习到更加精妙的图像作假技术，Discriminator则会成长为能以更高精度辨别真假的鉴定师。两者互相切磋、共同成长，这是**GAN**（Generative Adversarial

Network）这个技术的有趣之处。在这样的切磋中成长起来的Generator最终会掌握画出足以以假乱真的图像的能力（或者说有这样的可能）。

之前我们见到的机器学习问题都是被称为**监督学习**（supervised learning）的问题。这类问题就像手写数字识别一样，使用的是图像数据和教师标签成对给出的数据集。不过这里讨论的问题，并没有给出监督数据，只给了大量的图像（图像的集合），这样的问题称为**无监督学习**（unsupervised learning）。无监督学习虽然是很早之前就开始研究的领域（Deep Belief Network、Deep Boltzmann Machine等很有名），但最近似乎并不是很活跃。今后，随着使用深度学习的DCGAN等方法受到关注，无监督学习有望得到进一步发展。

8.5.3 自动驾驶

计算机代替人类驾驶汽车的自动驾驶技术有望得到实现。除了汽车制造商之外，IT企业、大学、研究机构等也都在为实现自动驾驶而进行着激烈的竞争。自动驾驶需要结合各种技术的力量来实现，比如决定行驶路线的路线计划（path plan）技术、照相机或激光等传感技术等，在这些技术中，正确识别周围环境的技术据说尤其重要。这是因为要正确识别时刻变化的环境、自由来往的车辆和行人是非常困难的。

如果可以在各种环境中稳健地正确识别行驶区域的话，实现自动驾驶可能也就没那么遥远了。最近，在识别周围环境的技术中，深度学习的力量备受期待。比如，基于CNN的神经网络SegNet[42]，可以像图8-25那样高精度地识别行驶环境。

图8-25中对输入图像进行了分割（像素水平的判别）。观察结果可知，在某种程度上正确地识别了道路、建筑物、人行道、树木、车辆等。今后若能基于深度学习使这种技术进一步实现高精度化、高速化的话，自动驾驶的实用化可能也就没那么遥远了。

图8-25 基于深度学习的图像分割的例子：道路、车辆、建筑物、人行道等被高精度地识别了出来（引用自文献[43]）

8.5.4 Deep Q-Network（强化学习）

就像人类通过摸索试验来学习一样（比如骑自行车），让计算机也在摸索试验的过程中自主学习，这称为**强化学习**（reinforcement learning）。强化学习和有“教师”在身边教的“监督学习”有所不同。

强化学习的基本框架是，代理（Agent）根据环境选择行动，然后通过这个行动改变环境。根据环境的变化，代理获得某种报酬。强化学习的目的是决定代理的行动方针，以获得更好的报酬（图8-26）。

图8-26中展示了强化学习的基本框架。这里需要注意的是，报酬并不是确定的，只是“预期报酬”。比如，在《超级马里奥兄弟》这款电子游戏中，让马里奥向右移动能获得多少报酬不一定是明确的。这时需要从游戏得分（获得的硬币、消灭的敌人等）或者游戏结束等明确的指标来反向计算，决定“预期报酬”。如果是监督学习的话，每个行动都可以从“教师”那里获得正确的评价。

在使用了深度学习的强化学习方法中，有一个叫作Deep Q-Network（通称**DQN**）[44]的方法。该方法基于被称为Q学习的强化学习算法。这里省略

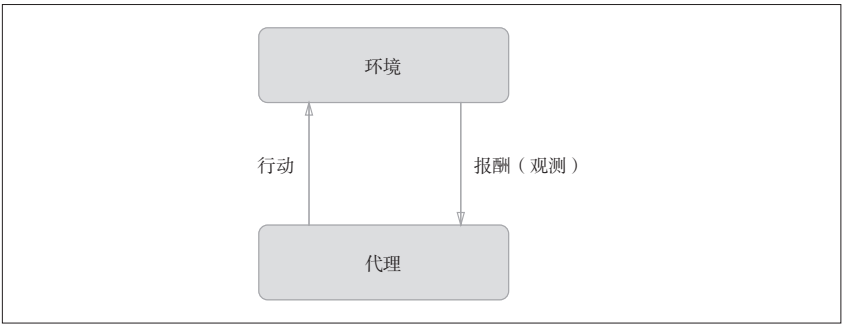

图8-26 强化学习的基本框架：代理自主地进行学习，以获得更好的报酬

Q学习的细节，不过在Q学习中，为了确定最合适的行动，需要确定一个被称为最优行动价值函数的函数。为了近似这个函数，DQN使用了深度学习（CNN）。

在DQN的研究中，有让电子游戏自动学习，并实现了超过人类水平的操作的例子。如图8-27所示，DQN中使用的CNN 把游戏图像的帧（连续4帧）作为输入，最终输出游戏手柄的各个动作（控制杆的移动量、按钮操作的有无等）的“价值”。

之前在学习电子游戏时，一般是把游戏的状态（人物的地点等）事先提取出来，作为数据给模型。但是，在DQN中，如图8-27所示，输入数据只有电子游戏的图像。这是DQN值得大书特书的地方，可以说大幅提高了DQN的实用性。为什么呢？因为这样就无需根据每个游戏改变设置，只要给DQN游戏图像就可以了。实际上，DQN 可以用相同的结构学习《吃豆人》、*Atari*等很多游戏，甚至在很多游戏中取得了超过人类的成绩。

人工智能AlphaGo[45]击败围棋冠军的新闻受到了广泛关注。这个AlphaGo技术的内部也用了深度学习和强化学习。AlphaGo学习了3000万个专业棋手的棋谱，并且不停地重复自己和自己的对战，积累了大量的学习经验。AlphaGo和DQN都是Google的Deep Mind公司进行的研究，该公司今后的研究值得密切关注。

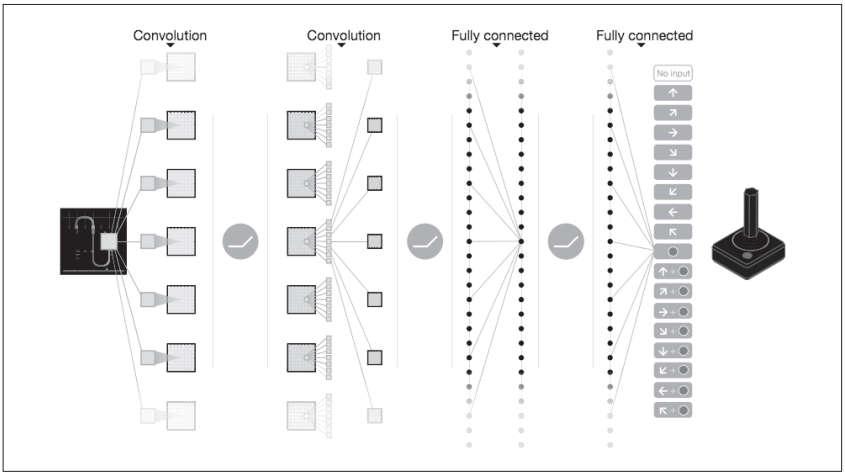

图8-27 基于Deep Q-Network学习电子游戏的操作。输入是电子游戏的图像，经过摸索试验，学习出让专业玩家都自愧不如的游戏手柄（操作杆）的操作手法（引用自文献[44]）

8.6 小结

本章我们实现了一个（稍微）深层的CNN，并在手写数字识别上获得了超过99%的高识别精度。此外，还讲解了加深网络的动机，指出了深度学习在朝更深的方向前进。之后，又介绍了深度学习的趋势和应用案例，以及对高速化的研究和代表深度学习未来的研究案例。

深度学习领域还有很多尚未揭晓的东西，新的研究正一个接一个地出现。今后，全世界的研究者和技术专家也将继续积极从事这方面的研究，一定能实现目前无法想象的技术。

感谢读者一直读到本书的最后一章。如果读者能通过本书加深对深度学习的理解，体会到深度学习的有趣之处，笔者将深感荣幸。

本章所学的内容

- 对于大多数的问题，都可以期待通过加深网络来提高性能。
- 在最近的图像识别大赛ILSVRC中，基于深度学习的方法独占鳌头，使用的网络也在深化。
- VGG、GoogLeNet、ResNet等是几个著名的网络。
- 基于GPU、分布式学习、位数精度的缩减，可以实现深度学习的高速化。
- 深度学习（神经网络）不仅可以用于物体识别，还可以用于物体检测、图像分割。
- 深度学习的应用包括图像标题的生成、图像的生成、强化学习等。最近，深度学习在自动驾驶上的应用也备受期待。