卷积神经网络的最新进展*

Jiuxiang Gu[1] Zhenhua Wang[2] Jason Kuen[2] Lianyang Ma[2] Amir Shahroudy[2]
Bing Shuai[2] Ting Liu[2] Xingxing Wang[2] Li Wang[2] Gang Wang[2]
Jianfei Cai[3] Tsuhan Chen[3]
[1] ROSE Lab, Interdisciplinary Graduate School, Nanyang Technological University, Singapore
[2]School of Electrical and Electronic Engineering, Nanyang Technological University, Singapore
[3] School of Computer Science and Engineering, Nanyang Technological University, Singapore

摘要

在过去的几年中，深度学习在视觉识别、语音识别和自然语言处理等各种问题上都取得了很好的效果。在不同类型的深度神经网络中，卷积神经网络的研究最为广泛。利用注释数据量的快速增长和图形处理器单元性能的巨大改进，卷积神经网络的研究迅速出现，并在各种任务上取得了最先进的结果。在本文中，我们提供了一个广泛的调查，在卷积神经网络方面最近的进展。我们从层设计、激活函数、损失函数、正则化、优化和快速计算等方面详细介绍了 CNN 的改进。此外，我们还介绍了卷积神经网络在计算机视觉、语音和自然语言处理中的各种应用。

1. 引言

卷积神经网络 (CNN) 是一个著名的深度学习架构，灵感来自生物的自然视觉感知机制。1959 年，Hubel & Wiesel 发现动物视觉皮层的细胞负责感受野的光探测。受此启发，日本福岛邦彦于 1980 年提出了新认知电子学，可以说是 CNN 的前身。1990 年 LeCun 等人发表了开创性的论文，建立了 CNN 的现代框架，并对其进行了改进。他们开发了一种名为 LeNet-5 的多层人工神经网络，可以对手写数字进行分类。像其他神经网络一样，LeNet-5 有多层，可以用反向传播算法进行训练。该方法可以获得原始图像的有效表示，使得无需预处理就能直接从原始像素中识别视觉模式成为可能。张等人的并行研究使用人工神经网络 (SIANN) 从图像中识别字符。然而，由于当时缺乏大量的训练数据和计算能力，他们的网络不能很好地处理更复杂的问题，例如大规模的图像和视频分类。自 2006 年以来，已经发展了许多方法来克服训练深度 cnn 所遇到的困难。最值得注意的是，Krizhevsky 等人提出了一个经典的 CNN 架构，并在图像分类任务上显示了对以前方法的显著改进。他们的方法的整体架构，即 AlexNet，与 LeNet-5 类似，但具有更深层次的结构。随着 AlexNet 的成功，人们提出了许多改进其性能的工作。其中，有代表性的工作有四个分别是 ZFNet, VGGNet, GoogleNet and ResNet。从架构演变来看，一个典型的趋势是网络越来越深，例如 2015 年 ILSVRC 冠军的 ResNet 比 AlexNet 深度约 20 倍，比 VGGNet 深度约 8 倍。通过增加深度，网络可以更好地逼近非线性增加的目标函数，得到更好的特征表示。但是，它也增加了网络的复杂性，使得网络更难以优化，更容易得到过拟合。在这个过程中，人们从各个方面提出了各种方法来解决这些问题。在本文中，我们试图对最近的进展进行全面的回顾，并进行一些深入的讨论。

*本文为论文 [1] 的中文翻译版，译者: 郭哲宏。由于内容较多，所以在 github 上持续更新，地址:https://github.com/vabook/paper

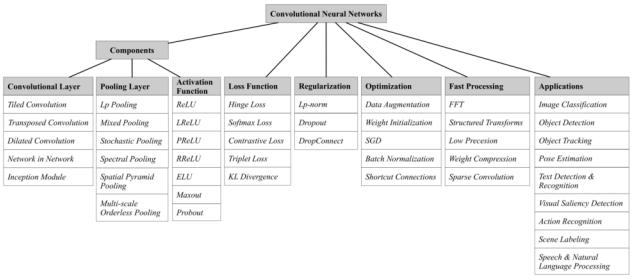


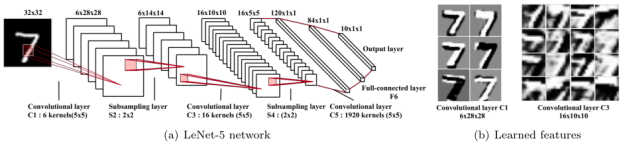
图 1. 卷积神经网络的层次结构分类

在下面的章节中，我们将确定与 CNN 相关的工作的大致类别。图1显示了本文的层次结构分类。我们首先在第 2 节中概述 CNN 的基本组件。然后在第 3 节介绍了 CNN 在不同方面的一些最新改进，包括卷积层、池化层、激活函数、损失函数、正则化和优化，并在第 4 节介绍了快速计算技术。接下来，我们在第 5 节中讨论了 CNN 的一些典型应用，包括图像分类、目标检测、目标跟踪、姿态估计、文本检测与识别、视觉显著性检测、动作识别、场景标记、语音和自然语言处理。最后，我们在第 6 节对本文进行总结。

2. CNN 基本组件

在文献中有很多 CNN 架构的变体。然而，它们的基本成分非常相似。以著名的 LeNet-5 为例，它由三种类型的层组成，即卷积层、池化层和全连接层。卷积层的目的是学习输入的特征表示。如图3(a)所示，卷积层由几个卷积核组成，这些卷积核用于计算不同的特征图。具体地说，特征图的每个神经元都连接到前一层邻近神经元的区域。这样的邻域被称为前一层神经元的接受域。新的特征图可以通过先将输入与学习过的核函数卷积，然后在卷积结果上应用一个元素级非线性激活函数得到。注意，要生成每个特性图，输入的所有空间位置都共享内核。通过使用几个不同的内核，得到了完整的特征映射。数学上，第 l 层的 k 个特征图 $z^l_{i,j,k}$ 中 (i,j) 位置的特征值是这样计算的:

$$z^l_{i,j,k} = w^l_k x^l_{i,j} + b^l_k \tag{1}$$



(a) LeNet-5 network (b) Learned features

图 2. LeNet-5 网络的结构

式中 w^l_k 和 b^l_k 分别为第 l 层第 k 个滤波器的权值向量和偏置项，$x^l_{i,j}$ 是以第 l 层 (i,j) 位置为中心的输入块。请注意内核 w^l_k 生成特性映射 $z^l_{:,:,k}$ 是共享的。这种权值共享机制具有降低模型复杂度、使网络更容易训练等优点。激活函数将非线性引入 CNN，这是多层网络检测非线性特征的理想方法。设 $a(\cdot)$ 表示非线性激活函数。卷积特征 $z^l_{i,j,k}$ 的激活值 $a^l_{i,j,k}$ 可以计算为:

$$a^l_{i,j,k} = a(z^l_{i,j.k}) \tag{2}$$

典型的激活函数是 sigmoid、tanh 和 ReLU。池化层的目的是通过降低特征图的分辨率来实现偏移不变性。它通常被放置在两个卷积层之间。池化层的每个特征图都与前一个卷积层对应的特征图相连接。将池函数表示为 $pool(\cdot)$，对于每个特征 $a^l_{:,:,k}$ 都有:

$$y^l_{i,j,k} = pool(a^l_{m,n,k}), \forall (m,n) \in R_{i,j} \tag{3}$$

其中 $R_{i,j}$ 是位置 (i,j) 附近的局部邻域。典型的池化操作是平均池化和最大池化。图 2(b) 是前两个卷积层学习到的数字 7 的特征图。第一卷积层的核被设计用于检测边缘和曲线等低级特征，而更高层的核被学习用于编码更抽象的特征。通过叠加几个卷积和池化层，我们可以逐渐提取更高层次的特征表示。

在几个卷积层和池化层之后，可能有一个或多个全连接层，目的是执行高级推理。它们将前一层的所有神经元与当前层的每一个神经元连接起来，生成全局语义信息。请注意，全连接层并不总是必要的，因为它可以被 1×1 卷积层取代。

CNN 的最后一层是输出层。对于分类任务，通常使用 softmax 操作符。另一种常用的方法是 SVM，它可以结合 CNN 的特征来解决不同的分类任务。让 θ 表示 CNN 的所有参数 (例如，权重向量和偏差项)。通过最小化定义在该任务上的适当损失函数，可以

获得特定任务的最佳参数。假设我们有 N 个期望的输入输出关系 $(x^{(n)}, y^{(n)}); n \in [1 \cdots N]$，其中 $x^(n)$ 为第 N 个输入数据，$y^(n)$ 为其对应的目标标签，$o^(n)$ 为 CNN 的输出。CNN 的损失值可以计算如下:

$$L = \frac{1}{N}\sum_{n=1}^{N} l(\theta; y^{(n)}, o^{(n)}) \tag{4}$$

训练 CNN 是一个全局优化问题。通过最小化损失函数，可以找到最佳的参数拟合集。随机梯度下降法是优化 CNN 网络的常用方法。

3. CNN 的改进

自从 AlexNet 在 2012 年成功以来，cnn 已经有了各种各样的改进。在本节中，我们将从六个方面描述 cnn 的主要改进: 卷积层、池化层、激活函数、损失函数、正则化和优化器。

3.1. 卷积层

基本神经网络中的卷积滤波器是一种广义线性模型 (GLM)。当潜在概念的实例是线性可分的时候，它很适合抽象概念。在这里我们介绍一些旨在提高其表现能力的工作。

3.1.1 平铺卷积 (Tiled Convolution)

神经网络中的权值共享机制可以大大减少参数的数量。然而，它也可能限制模型学习其他种类的不变性。Tiled CNN 是 CNN 的变种，拼贴并且复联特征图来学习旋转和规模不变的特征。在同一层中分别学习不同的核，通过对相邻单元进行平方根池化，可以隐式地学习复不变性。如图 3(b) 所示，卷积运算在每 k 个单元中进行，其中 k 是拼贴的大小，用来控制共享权值的距离。当拼贴的大小 k 为 1 时，每个 map 内的单位将具有相同的权重，tile CNN 将与传统 CNN 相同。在一篇论文中中，人们在 NORB 和 CIF AR-10 数据集上的实验表明，k = 2 的结果最好。Wang 等人发现 Tiled CNN 在小时间序列数据集上比传统 CNN 表现更好。

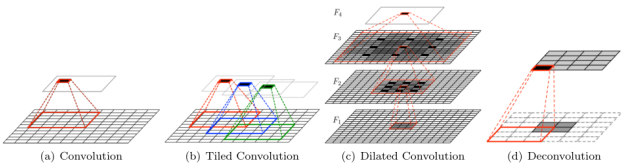


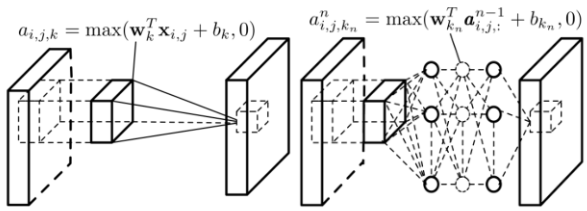
图 3. 四种卷积层

3.1.2 转置卷积 (Transposed Convolution)

转置卷积可以看作是相应的传统卷积的后向传递。它也被称为反卷积和分步卷积。为了与大多数文献一致，我们使用术语 “反卷积”。与传统的将多个输入激活连接到单个激活的卷积相反，反卷积将单个激活与多个输出激活相关联。图 3(d) 显示了在 4 × 4 输入上使用单位步幅和零填充的 3 × 3 核的反卷积运算。反卷积的步幅给出了输入特征图的膨胀因子。具体来说，反卷积首先对输入进行填充步长值的一个因子的上采样，然后对上采样的输入进行卷积运算。目前，反卷积被广泛应用于可视化、识别、定位、语义分割、视觉问答、超分辨率等领域。

3.1.3 扩张卷积 (Dilated Convolution)

Dilated CNN 是 CNN 的最新发展，它为卷积层引入了一个超参数。Dilated CNN 通过在过滤器元素之间插入零，可以增加网络接受域的大小，让网络覆盖更多的相关信息。这对于那些在做预测时需要很大接受域的任务来说是非常重要的。形式上，将信号 F 与核 k 大小为 r 的卷积的一维扩展卷积定义为 $(F *_l k)_t = \Sigma_\tau k_\tau F_{t-l\tau}$，其中 $*l$ 表示 l 扩展卷积。这个公式可以直接推广到二维扩张卷积。图 3(c) 显示了三个膨胀的卷积层的一个例子，其中膨胀因子 l 在每一层呈指数增长。中间的特征 F_2 由底部的特征图 F_1 通过施加一个扩展卷积产生，其中 F_2 中的每个元素都有一个接受域大小为 3×3。F_3 由 F_2 通过施加 2 扩张的卷积产生，其中 F_3 中的每个元素都有一个 $(2^3-1)\times(2^3-1)$ 的接受域。最上面的特征图 F_4 是由 F_3 通过 4 扩张卷积得到的，其中 F_4 中的每个元素的接受域为 $(2^4-1)\times(2^4-1)$。可以看出，F_{i+1} 中每个元素的感受场大小为 $(2^{i+2}-1)\times(2^(i+2)-1)$。Dilated cnn 在场景分割 [37]、机器翻译 [38]、语音



(a) Linear convolution layer (b) Mlpconv layer

图 4. 线性卷积层与 mlpconv 层的比较。

合成 [39] 和语音识别 [40] 等任务中取得了令人印象深刻的性能。

3.1.4 Network in Network

Network in Network(NIN) 是 Lin 等人提出的一种通用网络结构。它用微网络代替了卷积层的线性滤波器，如本文中的多层感知器卷积 (mlpconv) 层，使其能够近似于更抽象的潜在概念表示。NIN 的整体结构就是这些微网络的堆叠。图4显示了线性卷积层和 mlpconv 层之间的区别。形式上，卷积层的特征图 (具有非线性激活函数，如 ReLU[14]) 计算为

$$a_{i,j,k} = max(x_k^T x_{i,j} + b_k, 0) \tag{5}$$

其中 $a_{i,j,k}$ 是第 k 个特征图在 (i, j) 位置的激活值，$x_{i,j}$ 是以 (i, j) 位置为中心的输入，w_k 和 b_k 是第 k 个滤波器的权重向量和偏置项。作为比较，mlpconv 层执行的计算公式为

$$a^n_{i,j,k_n} = max(w^T_{k_n} a^{n-1}_{i,j,:} + b_{k_n}, 0) \tag{6}$$

其中 $n \in [1, N]$,N 是 mlpconv 层的层数，$a^0_{i,j,:}$ 等于 $x_{i,j}$。mlpconv 层在传统卷积层之后放置了 1×1 卷积。1×1 卷积相当于 ReLU 所继承的跨通道参数池操作。因此，mlpconv 层也可以看作是正常卷积层上级联的跨通道参数池化。最后，利用全局平均池对最后一层的特征图进行空间平均，并将输出向量直接输入到 softmax 层。与全连接层相比，全局平均池具有更少的参数，从而降低了过拟合风险和计算负荷。

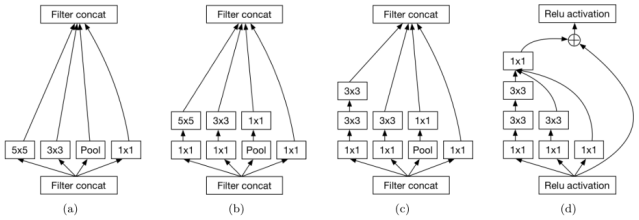


图 5. 不同版本的 Inception Module

3.1.5 Inception Module

Inception Module 由 Szegedy 等人引入，可以看作是 NIN 的逻辑顶点。他们使用不同尺寸的过滤器来捕捉不同尺寸的不同视觉模式，近似最优稀疏结构由 Inception Module 实现。具体而言，inception Module 实现由一个池化操作和三种类型的卷积操作组成 (见图5(b))，在 3×3 和 5×5 的卷积之前放置 1 × 1 convolutions 作为降维模块，这样可以在不增加计算复杂度的情况下增加 CNN 的深度和宽度。在初始模块的帮助下，网络参数可以大幅降低到 500 万，远低于 AlexNet(6000 万) 和 ZFNet(7500 万)

在他们后来的论文，为了找到具有相对适中计算成本的高性能网络，他们建议表示大小应该从输入到输出缓慢减少，并且在不损失表示能力的情况下，可以在低维嵌入上进行空间聚合。通过平衡每层滤波器的数量和网络的深度，可以达到网络的最佳性能。受 ResNet 的启发，他们最新的 inception-V4 结合了 inception 体系结构和快捷连接 (见图5(d))。他们发现，快捷连接可以显著加快初始网络的训练。在 ILSVRC2012 的验证数据集上，他们集成了 3 个残差和 1 个 Inception-v4 模型体系结构 (有 75 个可训练层) 可以达到 3.08% 的前 5 错误率。

3.2. 池化层

池化是 CNN 的一个重要概念。它通过减少卷积层之间的连接数来降低计算负担。在本节中，我们将介绍最近在 CNN 中使用的一些的池化方法。

3.2.1 L_P 池化

L_P 池化是一种生物启发的池化过程，以复杂细胞为模型。对其进行了理论分析，结果表明 L_P 池

化比最大池化具有更好的泛化能力。L_P 池化可以表示为

$$y_{i,j,k} = [\sum_{(m,n)\in R_{i,j}} (a_{m,n,k})^p]^{\frac{1}{p}} \tag{7}$$

其中，$y_{i,j,k}$ 是第 k 个特征图在 (i,j) 位置的池化操作的输出，$a_{m,n,k}$ 是第 k 个特征图的 $R_{i,j}$ 池化区域在 (m,n) 处的特征值。特别的，当 $p=1$ 时，L_p 等于平均池化，当 $p=\infty$ 时，，L_p 等于最大池化。

3.2.2 **混合池化** (Mixed Pooling)

Yu 等人受 random Dropout 和 DropConnect 的启发，提出了一种混合池化方法，即最大池化和平均池化相结合。混合池功能可以表述为如下:

$$y_{i,j,k} = \lambda \max_{(m,n)\in R_{i,j}} a_{m,n,k} + (1-\lambda)\frac{1}{|R_{i,j}|}\sum_{(m,n)\in R_{i,j}} a_{m,n,k} \tag{8}$$

其中 λ 是一个 0 或 1 的随机值，表示使用平均池或 Max 池的选择。在正向传播过程中，λ 被记录下来并用于反向传播操作。实验表明，混合池化能更好地解决过拟合问题，其性能优于最大池化和平均池化。

3.2.3 **随机池化** (Stochastic Pooling)

随机池化是一种受辍学启发的池化方法。随机池不像最大池那样在每个池域内选取最大值，而是根据多项式分布随机选取激活值，这保证了特征值的非最大激活值也可以被利用。具体来说，随机池首先通过归一化区域内的激活来计算每个区域 R_j 的概率 p，即 $p_i = a_i / \sum_{k\in R_j}(a_k)$。随后得到概率分布 $P(p1 \dots p|R_j|)$，我们可以从基于 p 的多项式分布中取样，在区域内选择一个位置 l，然后设置集合激活为 $y_j = a_l$，其中 $l \sim P(p1 \dots p|R_j|)$。与最大池化相比，随机池化可以避免由于随机成分而产生的过拟合。

3.2.4 **频谱池化** (Spectral Pooling)

频谱池化通过在频域裁剪输入表示来进行维数缩减。给定一个输入特征图 $x \in R^{mxm}$，假设所需的输出特性图的尺寸是 $h \times w$，频谱池化首先计算输入特征图的离散傅里叶变换 (DFT)，然后裁剪通过维持频率表示只有中央 h×w 频率的子矩阵，最后利用反转 DFT 将近似值映射回空间域。与最大池化相比，频谱池化的线性低通滤波操作可以在相同的输出维数下保留更多的信息。同时，它也不受其他池化方法输出图维数急剧下降的影响。更重要的是，频谱池化的过程是通过矩阵截断实现的，这使得它能够在使用 FFT 卷积核的 CNN 中实现，且计算成本很小。

3.2.5 **空间金字塔池化** (Spatial Pyramid Pooling)

空间金字塔池化 (SPP) 是由 He 等人引入的。SPP 的主要优点是，它可以生成固定长度的表示，而不管输入大小。SPP 池化将特征图输入到与图像大小成比例的局部空间箱中，从而得到固定的箱数量。这与之前的深度网络中的滑动窗口池不同，滑动窗口的数量取决于输入的大小。通过将最后一个池化层替换为 SPP，他们提出了一个新的 SPP 网络，可以处理不同大小的图像。

3.2.6 **多尺度无条理池化** (Multi-scale Orderless Pooling)

Gong 等人利用多尺度无序池 (MOP) 来改善神经网络的不变性，同时不降低其判别能力。他们提取了整个图像和多个尺度的局部块的深度激活特征。整个图像的激活与之前的 CNN 相同，目的是捕获全局空间布局信息。通过 VLAD 编码对局部块的激活进行聚合，目的是获取更多局部的、细粒度的图像细节，并增强图像的不变性。通过连接全局激活和局部块激活的 VLAD 特征，得到新的图像表示。

3.3. 激活函数

适当的激活函数可以显著提高 CNN 对某一任务的性能。在本节中，我们将介绍最近在 CNN 中使用的激活函数

3.3.1 ReLU

Rectified linear unit (ReLU) 是最引人注目的非饱和活化函数之一。ReLU 激活函数定义为:

$$a_{i,j,k} = \max(z_{i,j,k}, 0) \tag{9}$$

其中 $z_{i,j,k}$ 是激活函数在第 k 个通道的 (i, j) 处的输入。ReLU 是一个分段线性函数，将负的部分修剪为零，保留正的部分 (见图6(a))。ReLU 的简单 max(·) 运算使得它的计算速度比 sigmoid 或 tanh 激活函数快得多，而且它还能诱导隐藏单元的稀疏性，使网络易于获得稀疏表示。研究表明，即使不需要预先训练，使用 ReLU 也可以有效地训练深度网络。尽管 ReLU 在 0 处的不连续性可能会损害反向传播的性能，但许多研究表明，ReLU 比 sigmoid 和 tanh 激活函数更有效。

3.3.2 Leaky ReLU

ReLU 单元的一个潜在缺点是，当该单元不活动时，它的梯度为零。这可能会导致最初不活动的单位永远不会活动，因为基于梯度的优化不会调整它们的权重。此外，由于恒定的零梯度，它可能会减慢训练过程。为了缓解这个问题，Mass 等人引入了 Leaky ReLU (LReLU)，定义为:

$$a_{i,j,k} = \max(z_{i,j,k}, 0) + \lambda \min(z_{i,j,k}, 0) \tag{10}$$

其中 是 (0,1) 范围内的预定义参数。与 ReLU 相比，Leaky ReLU 压缩负的部分，而不是将其映射到常量零，这使得它给一个小的非零梯度，当这个单元不活动。

3.3.3 Parametric ReLU

与其在 leleaky ReLU 中使用预定义的参数，例如式10中的 λ，He 等人提出了参数整流线性单元 (PReLU)，它自适应地学习整流器的参数以提高精度。在数学上，PReLU 函数定义为:

$$a_{i,j,k} = \max(z_{i,j,k}, 0) + \lambda_k \min(z_{i,j,k}, 0) \tag{11}$$

其中 λ_k 是第 k 个通道的学习参数。由于 PReLU 只引入非常少量的额外参数，额外参数的数量与整个网络的通道数量相同，因此不存在额外的过拟合风险，额外的计算代价可以忽略。它也可以通过反向传播的方法同时与其他参数进行训练。

3.3.4 Randomized ReLU

Leaky ReLU 的另一种变体是随机泄漏整流线性单元 (RReLU)。在 RReLU 中，负部分的参数在训练时从均匀分布中随机采样，然后在测试时固定 (见图6(c))。形式上，RReLU 函数定义为:

$$a^{(n)}_{i,j,k} = \max(z^{(n)}_{i,j,k}, 0) + \lambda^{(n)}_k \min(z^{(n)}_{i,j,k}, 0) \tag{12}$$

其中 $z^{(n)}_{i,j,k}$ 是第 n 个例子的第 k 个通道在 (i, j) 位置激活函数的输入，$\lambda^{(n)_k}$ 为对应的采样参数，$a^{(n)}_{i,j,k}$ 为对应的输出。由于它的随机特性，可以减少过拟合。Xu 等人也评价了 ReLU、LReLU、PReLU 和 RReLU 在标准图像分类任务中的作用，并得出结论: 在校正激活单元中加入负部分的非零斜率可以持续提高性能。

3.3.5 ELU

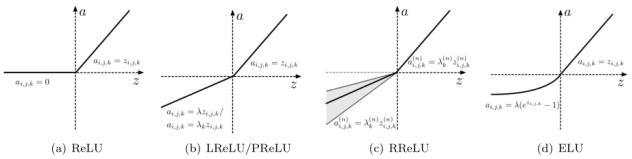
Clevert 等人引入了指数线性单元 (ELU)，使深度神经网络的学习速度更快，并导致更高的分类精度。与 ReLU、LReLU、PReLU 和 RReLU 一样，ELU 通过将正数部分设置为辨别部分来避免梯度渐变消失问题。与 ReLU 相比，ELU 有利于快速学习的负数部分。相对于 LReLU、PReLU、RReLU 也有不饱和的负数部分，ELU 采用饱和函数作为负数部分。由于饱和函数在失活时将减少单元的变化，使 ELU 对噪声更具有鲁棒性。ELU 的函数定义为:

$$a_{i,j,k} = \max(z_{i,j,k}, 0) + \min(\lambda(e^{z_{i,j,k}} - 1), 0) \tag{13}$$

其中 λ 是一个预定义的参数，以控制 ELU 饱和负输入的值。

3.3.6 Maxout

Maxout 是一个可选非线性函数，它在每个空间位置上具有多个通道的最大响应。maxout 函数定义为 $a_{i,j,k} = \max_{k \in [i,K]} z_{i,j,k}$，其中在 $z_{i,j,k}$ 是特征图



(a) ReLU (b) LReLU/PReLU (c) RReLU (d) ELU

图 6. 四种激活函数

的第 k 个通道。值得注意的是，maxout 享有 ReLU 的所有好处，因为 ReLU 实际上是 maxout 的一种特殊情况，例如，$\max(w_1^T x + b_1, w_2^T x + b_2)$，其中 w1 是一个零向量，b1 是零。此外，maxout 特别适合于 Dropout 的训练

3.3.7 Probout

Springenberg 等人提出了一种名为 probout 的 maxout 的概率变体。它们用概率抽样程序代替了 maxout 中的最大运算。具体来说，他们首先定义每个 k 线性单位的概率为:$p_i = e^{\lambda z_i} / \sum_{i=1}^{k} e^{\lambda z_j}$，其中 λ 是控制分布方差的超参数。然后，他们根据多项分布从 k 个单位中选出一个 $\{p1 \dots pk\}$，并设置激活值为所选单位的值。为了与 dropout 相结合，他们实际上重新定义了概率:

$$\hat{p}_0 = 0.5, \hat{p}_i = e^{\lambda z_j} / (2. \sum_{j=1}^{k} e^{\lambda z_j}) \tag{14}$$

然后将激活函数采样为

$$a_i = \begin{cases} 0, & \text{if i} = 0 \\ z_i, & \text{else} \end{cases} \tag{15}$$

其中 $i \sim$ 多项式 $\{\hat{p}_0, ..., \hat{p}_k\}$。Probout 可以在保持最大输出单元的理想性质和改善其不变性之间取得平衡。然而，在测试过程中，由于附加的概率计算，probout 的计算成本比 maxout 高。

3.4. 损失函数

为特定的任务选择适当的损失函数是很重要的。在本节中我们介绍了四个具有代表性的损失函数:Hinge loss, Softmax loss, Contrastive loss, Triplet loss。

3.4.1 Hinge loss

Hinge loss 通常用于训练大幅度分类器，如支持向量机 (SVM)。多类支持向量机的 hinge loss 定义在(16)中，其中 w 是分类器的权重向量，$y^{(i)} \in [1 \dots K]$ 表示它在 K 个类中正确的类标签。

$$L_{hinge} = \frac{1}{N} \sum_{i=1}^{N} \sum_{j=1}^{K} [\max(0, 1 - \delta(y^{(i)}, j) w^T x_i)]^p \tag{16}$$

其中 $\delta(y^{(i)}, j) = 1$ $y^{(i)} = j$，$\delta(y^{(i)}, j) = 1$，注意如果 $p = 1$ 等式(16)为 Hinge-Loss(L1-Loss)，如果 $p = 2$, 则为 Hinge-Loss(L2-Loss)。L2-Loss 与 L1-Loss 比较是可微的，并且对违反边际的点施加更大的损失。有人研究并比较了 softmax 和 L2-SVMs 在深度网络中的性能。MNIST 的结果表明 L2-SVM 优于 softmax。

3.4.2 Softmax Loss

Softmax Loss 是一种常用的损失函数,实质上是多项 logistic loss 和 Softmax 的组合。给定一个训练集 $\{(x^{(i)}, y^{(i)}); i \in 1 ..., N, y^{(i)} \in 1 ..., K\}$，其中 $x^{(i)}$ 为第 i 个输入图像块,$y^{(i)}$ 为其 K 个类中的目标类标签。对第 i 个输入的第 j 类的预测用 softmax 函数进行变换:$p_j^{(i)} = e^{z_j^{(i)}} / \sum_{l=1}^{K} e^{z_l^{(i)}}$，其中 $z_j^{(i)}$ 通常是密连接层的激活，因此 $z_j^{(i)}$ 可以写成 $z_j^{(i)} = w_j^T a^{(i)} + b_j$。Softmax 将预测转化为非负值，并将其归一化，得到类的概率分布。这种概率预测用于计算多项 logistic loss，即 softmax loss，如下所示:

$$L_{softmax} = -\frac{1}{N} [\sum_{i=1}^{N} \sum_{j=1}^{K} \{y^{(i)} = j\} log p_j^{(i)}] \tag{17}$$

最近，Liu 等人提出了大幅度 Softmax(L-Softmax)loss，该损失在输入特征向量 $a^{(i)}$ 和权重矩阵的第 j 列 w_j 之间的角度 θ_j 引入了一个角幅度。L-Softmax 损耗的预测 $p_j^{(i)}$ 定义为:

$$p_j^{(i)} = \frac{e^{\|w_j\| \left\|a^{(i)}\right\| \psi(\theta_j)}}{e^{\|w_j\| \left\|a^{(i)}\right\| \psi(\theta_j)} + \sum_{l \neq j} e^{\|w_j\| \left\|a^{(i)}\right\| \cos(\theta_l)}} \tag{18}$$

$$\psi(\theta_j) = (-1)^k \cos(m\theta_j) - 2k, \theta_j \in [k\pi/m, (k+1)\pi/m] \tag{19}$$

其中 $k \in [0, m-1]$ 是一个整数，m 控制类之间的幅度。当 m = 1 时，L-Softmax loss 减小到原来的 softmax loss。通过调整类间的幅度 m，定义一个相对困难的学习目标，可以有效避免过拟合。他们验证了 L-Softmax 对 MNIST、CIFAR-10 和 CIFAR-100 的有效性，并发现 L-Softmax 的损耗性能优于原 softmax。

3.4.3 Contrastive Loss

Contrastive Loss 通常用于训练 Siamese 网络，这是一种弱监督方案，用于从标记为匹配或非匹配的成对数据实例中学习相似性度量。已知第 i 对数据 $(x_\alpha^{(i)}\ x_\beta^{(i)})$，设 $(z_\alpha^{(i,l)}\ z_\beta^{(i,l)})$ 表示第 l 层 $(l \in [1\ \cdots\ l])$ 对应的输出对。在一篇论文中它们将图像对通过两个相同的 CNN，并将最后一层的特征向量提供给 cost 模块。他们训练样本使用的对比损失函数为:

$$L_{contrastive} = \frac{1}{2N}\sum_{i=1}^{N}(y)d^{i,L}+(1-y)max(m-d^(i,L),0) \tag{20}$$

其中 $d(i,L) = \left\|z_\alpha^{i,L} - z_\beta^{i,L}\right\|_2^2$，m 是影响非匹配对的边界参数。如果 $(xx_\alpha^{(i)}\ x_\beta^{(i)})$ 是一个匹配对，则 y = 1。否则，y = 0。Lin 等人发现，当对所有对进行微调时，这种单一的边际损失函数会导致检索结果的急剧下降。同时，仅对非匹配对进行微调时，性能得到了较好的保留。这表明丢失是由 loss 函数中对匹配对的处理造成的。虽然仅对非匹配对的召回率是稳定的，但对匹配对的处理是召回率下降的主要原因。为了解决这一问题，他们提出了一个双幅度损失函数，该函数增加了另一个幅度参数来影响匹配对。不是计算最后一层的损失，而是为每一层 l 定义它们的对比损失，并同时执行单个层的损失的反向传播。它被定义为:

$$\begin{aligned} L_{contrastive} = \frac{1}{2N}\sum_{i=1}^{N}\sum_{i=1}^{L}(y)max(d^{i,L} - m_{1,0}) \\ +(1-y)max(m_2 - d^{(i,L)},0) \end{aligned} \tag{21}$$

在实践中，他们发现这两个边缘参数可以设置为相等 $(m1 = m2 = m)$，可以通过采样匹配和非匹配图像对的分布来学习。

3.4.4 Triplet Loss

Triplet Loss 考虑每个损失函数的三个实例。三元组单元 $(x_a^{(i)}, x_p^{(i)}, x_n^{(i)})$ 通常包含一个锚实例 $x_a^{(i)}$ 以及来自 $x_a^{(i)}$ 同一类正向实例 $x_p^{(i)}$ 和一个反向实例 $x_n^{(i)}$。让 $(z_a^{(i)}, z_p^{(i)}, z_n^{(i)})$ 表示为三元体单元的特征,Triplet Loss 被定义为:

$$L_{triplet} = \frac{1}{N}\sum_{i=1}^{N}\max\{d_{(a,p)}^{(i)} - d_{(a,n)}^{(i)} + m, 0\} \tag{22}$$

其中 $d_{(a,p)}^{(i)} = \left\|z_a^{(i)} - z_p^{(i)}\right\|_2^2 = \left\|z_a^{(i)} - z_n^{(i)}\right\|_2^2$。Triplet Loss 的目的是使锚点与正向之间的距离最小，而使负向与锚点之间的距离最大。

但在某些特殊情况下，随机选取的锚点样本可能会判断错误。例如，当 $d_{(n,p)}^{(i)} < d_{(a,p)}^{(i)} < d_{(a,n)}^{(i)}$ 时，三元组损失可能仍然为零。因此，在反向传播过程中，将忽略三元组单元。Liu 等人提出耦合簇损失 (Coupling Clusters, CC) 来解决这一问题。耦合簇损失函数是在正集和负集上定义的，而不是使用三元组单元。将随机选取的锚点替换为聚类中心，使正集的样本聚在一起，而负集的样本相对远离，比原来的三元组损失更可靠。耦合簇损失函数定义为:

$$L_{cc} = \frac{1}{N^p}\sum_{i=1}^{N^p}\frac{1}{2}\max\{\left\|z_p^{(i)} - c_p\right\|_2^2 - \left\|z_n^{(*)} - c_p\right\|_2^2 + m, 0\} \tag{23}$$

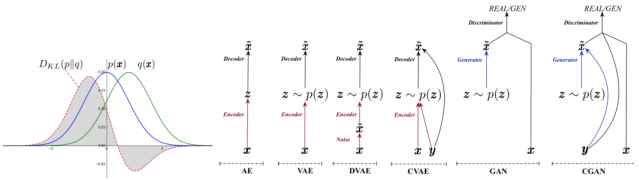
其中 N^p 是每个集合的样本数，$z_n^{(*)}$ 是 $x_n^{(*)}$ 的特征表示，$x_n^{(*)}$ 是离估计中心点 $c_p = (\sum_i^{N^p} z_p^{(i)})/N^p$ 最近的负样本。三元组损失及其变体被广泛应用于各种任务中，包括再识别、验证和图像检索。

3.4.5 Kullback-Leibler Divergence

Kullbeck-Leibler Divergence(KLD) 是两个概率分布 $p(x)$ 和 $q(x)$ 在同一个离散变量 x 上差异的非对称度量 (见图7(a))。从 $q(x)$ 到 $p(x)$ 的 KLD 定义为:

$$D_{KL}(p\|q) = -H(p(x)) - E_p[\log q(x)] \tag{24}$$

$$= \sum_x p(x)\log p(x) - \sum_x p(x)\log q(x) = \sum_x p(x)\log\frac{p(x)}{q(x)} \tag{25}$$



(a) KL divergence (b) Autoencoders variants and GAN variants

图 7. KLD

其中 $H(p(x))$ 是 p(x) 的香农熵，$Ep(\log q(x))$ 是 $p(x)$ 和 $q(x)$ 的交叉熵。

KLD 被广泛用于各种自动编码器 (AEs) 的目标函数中作为信息损失的度量。自动编码器的著名变种包括稀疏自动编码器，去噪声自动编码器和变分自动编码器 (VAE)。VAE 通过贝叶斯推理解释潜在表征。它由两部分组成: 编码器将数据样本 x 压缩为潜在表征 $z \sim q_\phi(z|x)$; 以及一个解码器，该解码器将这样的表示映射回尽可能接近输入的数据空间 $\hat{x} \sim p_\theta(x|z)$。其中 ϕ 为编码器参数，θ 为解码器参数。在一篇论文中，VAEs 试图最大化 $\log p(x|\phi,\theta)$ 的对数似然的变分下界:

$$L_{vae} = E_{z\sim q_\phi(z|x)}[\log p_\theta(x|z)] - D_{KL}(q_\phi(z|x)||p(z)) \tag{26}$$

其中第一项是重建成本，KLD 项强制对分布 $q\phi(z|x)$ 施加先验 $p(z)$。通常 $p(z)$ 是标准正态分布，离散分布，或一些具有几何解释的分布。继最初的 VAE 之后，许多变种已经被提出。条件 VAE(CVAE) 从条件分布中使用 $\hat{x} \sim p_\theta(x|y,z)$ 生成样本。去噪 VAE(DVAE) 从损坏的输入 $\hat{x}$ 中恢复原始输入 x。

Jensen-Shannon Divergence (JSD) 是 KLD 的对称形式。它衡量了 $p(x)$ 和 $q(x)$ 之间的相似性:

$$D_{JS}(p||q) = \frac{1}{2}D_{KL}\left(p(x)||\frac{p(x)+q(x)}{2}\right) + \frac{1}{2}D_{KL}\left(q(x)||\frac{p(x)+q(x)}{2}\right) \tag{27}$$

通过最小化 JSD，我们可以使 $p(x)$ 和 $q(x)$ 两个分布尽可能接近。JSD 已成功应用于生成对抗网络 (GANs)。与直接建模 x 和 z 之间关系的 VAEs 不同，GANs 被明确地设置为优化生成任务。GANs 的目标是找到能在真实数据和生成数据之间给出最佳区分的鉴别器 D，同时鼓励生成器 G 拟合真实数据分布。鉴别器 D 和生成器 G 之间的最小-最大博弈被以下目标函数形式化:

$$\min_G \max_D \mathcal{L}_{gan}(D,G) = E_{x\sim p(x)}[\log D(x)] + E_{z\sim q(z)}[\log(1-D(G(z)))] \tag{28}$$

最初的 GAN 论文表明，对于固定生成器 G^*，我们有最优鉴别器 $D^*G(x) = \frac{p(x)}{p(x)+q(x)}$。那么方程(28)等价于使 JSD 在 $p(x)$ 和 $q(x)$ 之间最小化。如果 G 和 D 有足够的容量，分布 $q(x)$ 收敛于 $p(x)$。与条件式 VAE 一样，条件式 GAN(CGAN) 也接收额外的信息 y 作为输入，以生成约束 y 的样本。在实践中，众所周知，GANs 在训练时是不稳定的。

3.5. Regularization

在深度神经网络中，过拟合是一个不可忽视的问题，通过正则化可以有效地降低过拟合。在下面的小节中，我们将介绍一些有效的正则化技术:ℓ_p-norm，Dropout 和 DropConnect。

3.5.1 ℓ_p-norm Regularization

正则化通过增加额外的项来修改目标函数，以减少模型的复杂性。形式上，如果损失函数为 $\mathcal{L}(\theta,x,y)$，则正则化损失为:

$$E(\theta,x,y) = \mathcal{L}(\theta,x,y) + \lambda R(\theta) \tag{29}$$

其中 $R(\theta)$ 是正则化项，λ 是正则化强度。

ℓ_p-norm 正则化函数通常采用 $R(\theta) = \sum_j ||\theta_j||_p^p$，当 $P \geq 1$ 时，ℓ_p-norm 是凸的，使优化更容易，使该函数具有吸引力。对于 p=2，ℓ_p-norm 正则化通常被称为权重衰减。ℓ_p-norm 的一个更有原则的替代方案是 Tikhonov 正则化，它奖励对输入噪声的不变性。当 p<1 时，ℓ_p-norm 正则化更多地利用了权值的稀疏性，但对非凸函数有一定的作用。

3.5.2 Dropout

Dropout 首先是由 Hinton 等人引入的，已经被证明在减少过拟合方面非常有效。在他们的论文中，他们对完全连接的层应用 Dropout。Dropout 的输出是 $y = r*a(W^T\boldsymbol{x})$，其中 $\boldsymbol{x} = [x_1, x_2 \ldots, x_n]^T$ 是全

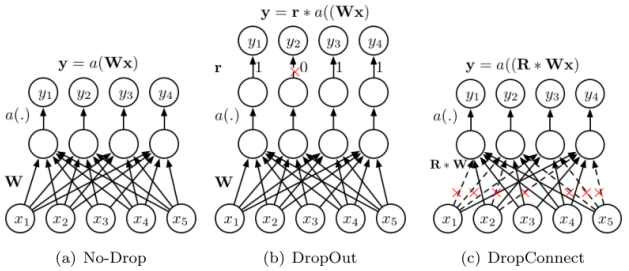


图 8. 三种网络的描述

连接层的输入，$W \in R^{n \times d}$ 是一个权值矩阵，$\boldsymbol{r}$ 是一个大小为 d 的二进制向量，其元素是由参数为 p 的伯努利分布独立得出的，即 $r_i \sim Bernoulli(p)$。Dropout 可以防止网络过于依赖任何一个神经元 (或任何一个小的神经元组合)，并且可以迫使网络在缺乏特定信息的情况下保持准确。已经提出了几种改进 Dropout 的方法。Wang 等人提出了一种快速 Dropout 方法，该方法可以通过采样或积分高斯近似来进行快速 Dropout 训练。Ba 等人 [91] 提出了一种自适应 Dropout 方法，其中每个隐藏变量的 Dropout 概率使用与深度网络共享参数的二进制网络来计算。在论文中，他们发现在 1×1 卷积层之前使用标准的 Dropout 一般会增加训练时间，但不能防止过拟合。因此，他们提出了一个名为 Spatial-Dropout 的新 Dropout 方法，它将 Dropout 值扩展到整个特征图。这种新的 Dropout 方法尤其适用于训练数据量较小的情况。

3.5.3 DropConnect

DropConnect 将 Dropout 的理念更进一步。DropConnect 不是将神经元的输出随机设置为零，而是将权重矩阵 $\boldsymbol{W}$ 的元素随机设置为零。DropConnect 的输出由 $y = a((R * W)x)$ 给出，其中 $R_{i,j} \sim Bernoulli(p)$。此外，在训练过程中，偏差也被掩盖了。图8说明了 No-Drop，Dropout 和 DropConnect 网络之间的差异。

3.6. 最优化

在本小节中，我们将讨论优化 CNNs 的一些关键技术。

3.6.1 数据增强

深层的 CNNs 特别依赖于大量训练数据的可用性。与 CNNs 中涉及的参数数量相比，一个缓解数据相对稀缺的优雅解决方案是数据增强。数据增强包括将可用数据转换为新数据而不改变其性质。流行的增强方法包括简单的几何变换，如采样、镜像、旋转、移动和各种光度变换。Paulin 等人提出了一种贪婪策略，从一组候选变换中选择最佳变换。然而，他们的策略涉及大量的模型再训练步骤，当候选转换的数量很大时，这在计算上是昂贵的。Hauberg 等人提出了一种优雅的方法，通过随机生成微分态来进行数据增强。Xie 等人和 Xu 等人提供了从互联网收集图像的额外方法，以改善细粒度识别任务中的学习。

3.6.2 权重初始化

深层的 CNN 具有大量的参数，其损失函数是非凸的，这使得它很难训练。为了在训练中实现快速收敛，避免消失梯度问题，适当的网络初始化是最重要的先决条件之一。偏移参数可以初始化为零，而权值参数应谨慎初始化，以打破同一层隐藏单元之间的对称性。如果网络没有正确初始化，例如，每一层将其输入缩放为 k，最终的输出将缩放原始输入为 k^L，其中 L 是层数。在这种情况下，$k > 1$ 的值会导致输出层的值非常大，而 $k < 1$ 的值会导致输出值递减，并且呈梯度化。Krizhevsky 等人将其网络的权值从标准差为 0.01 的零均值高斯分布初始化，并将第 2、4、5 个卷积层以及所有全连接层的偏差项设为常数 1。另一种著名的随机初始化方法是 “Xavier”。他们从一个均值为零、方差为 $\frac{2}{n_{in}+n_{out}}$ 的高斯分布中选择权值，其中 n_{in} 为输入该分布的神经元数量，而 n_{out} 为输出结果的神经元数量。因此 “Xavier” 可以根据输入输出神经元的数量自动确定初始化规模，并通过多层将信号保持在一个合理的值范围内。它在 Caffe 中的一个变体仅使用了 n_{in}-only，这使得它更容易实现。“Xavier” 初始化方法后来被扩展，以考虑校正的非线性，其中他们推导了一个鲁棒的初始化方法，特别考虑 ReLU 的非线性。他们的方法

允许训练极深的模型使其收敛，而 “Xavier” 方法却不能。

Saxe 等人独立地表明，正交矩阵初始化对于线性网络比高斯初始化更有效，对于非线性网络也同样有效。Mishkin 等人将其扩展为一个迭代过程。具体来说，它提出了一种层序单位方差处理方案，可以将其视为只对第一个小批处理执行的标准正交初始化与批标准化相结合 (见 3.6.4 节)。它类似于批量归一化，因为它们都采用了单位方差归一化处理。不同的是，它使用正交归一化来初始化权重，这有助于有效地去相关层活动。这种初始化技术被应用了，而且性能显著提高。

3.6.3 随机梯度下降

反向传播算法是使用梯度下降更新参数的标准训练方法。许多梯度下降优化算法已经被提出。标准梯度下降算法将目标 $\mathcal{L}(\theta)$ 的参数 θ 更新为 $\theta_{t+1} = \theta_t - \eta\nabla_\theta \mathrm{E}[\mathcal{L}(\theta_t)]$，其中 $\mathrm{E}[\mathcal{L}(\theta_t)]$ 是 $\mathcal{L}(\theta_t)$ 在整个训练集上的期望，η 是学习率。随机梯度下降 (SGD) 不是计算 $\mathcal{L}(\theta_t)$，而是从训练集中随机选取的单个样例 $(x^{(t)}, y^{(t)})$ 估计梯度。

$$\theta_{t+1} = \theta_t - \eta_t \nabla_\theta \mathcal{L}(\theta_t; x^{(t)}, y^{(t)}) \tag{30}$$

在实践中，SGD 中的每个参数更新都是根据一个小批处理计算的，而不是单个示例。这有助于减少参数更新中的方差，并能促使更稳定的收敛。收敛速度由学习率 η_t 控制。然而，小批量 SGD 并不能保证良好的收敛性，仍然存在一些需要解决的挑战。首先，选择一个合适的学习率并不容易。一种常见的方法是使用一个恒定的学习率，在初始阶段给出稳定的收敛，然后随着收敛速度的减慢而降低学习率。另外，学习速率计划被提出用于在训练过程中调整学习率。为了使当前梯度更新依赖于历史批次和加速训练，动量被提出用于在相关方向累加速度矢量。经典的动量更新公式是这样定义的:

$$v_{t+1} = \gamma v_t - \eta_t \nabla_\theta \mathcal{L}(\theta_t; x^{(t)}, y^{(t)}) \tag{31}$$

$$\theta_{t+1} = \theta_t + v_{t+1} \tag{32}$$

其中 v_{t+1} 是当前的速度矢量，γ 是动量项，通常设置为 0.9。Nesterov 动量是在梯度下降优化中使用动量的另一种。

$$v_{t+1} = \gamma v_t - \eta_t \nabla_\theta \mathcal{L}(\theta_t + \gamma v_t; x^{(t)}, y^{(t)}) \tag{33}$$

与经典动量先计算当前梯度，然后向更新累加梯度的方向移动相比，Nesterov 动量先向先前累积梯度 γv_t 的方向移动，计算梯度，然后进行梯度更新。这种预期的更新可以防止优化进行得太快，从而获得更好的性能。

并行 SGD 方法改进了 SGD，使之适合于并行、大规模机器学习。与标准 (同步)SGD 不同的是，在标准 (同步)SGD 中，如果一台机器太慢，训练就会被延迟，这个并行化方法使用异步机制，因此除了最慢的机器上的优化外，没有其他优化会被延迟。Jeffrey Dean 等人使用另一种称为 Downpour SGD 的异步 SGD 程序来加快具有多个 cpu 的集群上的大规模分布式训练进程。也有一些工作是使用多个 GPU 的异步 SGD。Paine 等人基本上将异步 SGD 与 gpu 结合起来，与在单机上训练相比，可以将训练时间缩短几倍。Zhuang 等人也使用多个 GPU 异步计算梯度并更新全局模型参数，在 4 个 GPU 上实现的加速比在单个 GPU 上的快 3.2 倍。

注意，SGD 方法可能不会促使收敛。当性能停止提升时，训练过程可以终止。对过度训练的一个流行的补救方法是使用早期停止，在训练期间，基于验证集的性能停止优化。为了控制训练过程的持续时间，可以考虑不同的停止标准。例如，训练可能执行固定的 epoch 数，或直到达到预定义的训练错误。停止策略需要谨慎操作，在提高网络泛化能力和避免过拟合的前提下，适当的停止策略应该让训练过程继续进行。

3.6.4 批量归一化

数据归一化通常是数据预处理的第一步。全局数据归一化将所有数据转化为零均值和单位方差。但是，当数据流经深层网络时，输入到内层的分布会发生变化，从而失去网络的学习能力和准确性。Ioffe 等人提出了一种有效方法称为批量归一化 (Batch

Normalization, BN) 来部分缓解这一现象。它通过一个归一化步骤来解决所谓的协变量偏移问题，该步骤固定了输入层的均值和方差，其中均值和方差的估计是在每一个小批次后计算的，而不是整个训练集。假设待规格化的层有一个 d 维输入，即 $x = [x_1, x_2, ..., x_d]^T$。我们首先将第 k 维归一化如下:

$$\hat{x}_k = (x_k - \mu_{\mathcal{B}})/\sqrt{\delta_{\mathcal{B}}^2 + \epsilon} \tag{34}$$

其中 $\mu_{\mathcal{B}}$ 和 $\delta_{\mathcal{B}}^2$ 分别为小批量的均值和方差，δ 是一个常数值。为了增强表示能力，将归一化输入 $\hat{x}_k$ 进一步转换为:

$$y_k = BN_{\gamma,\beta}(x_k) = \gamma\hat{x}_k + \beta \tag{35}$$

其中 γ 和 β 是学习参数。与全局数据归一化相比，批量归一化具有许多优点。首先，它减少了内部协变量偏移。其次，BN 减少了梯度对参数尺度或其初始值的依赖，从而对网络中的梯度流动产生有益的影响。这使得使用更高的学习率，而没有出现分歧的风险。此外，BN 正则化了模型，从而减少了 Dropout 的需要。最后，BN 使使用饱和非线性激活函数而不陷入饱和模型成为可能。

3.6.5 快捷连接

如上所述，通过规一初始化和 BN 可以缓解深度神经网络的消失梯度问题。尽管这些方法成功地防止了深度神经网络的过拟合，但它们也给网络的优化带来了困难，导致其性能比浅层网络差。这种深层神经网络所遭受的优化问题被视为退化问题。

受长短期记忆 (LSTM) 网络的启发，该网络使用门函数来确定有多少神经元的激活值要转换或只是通过。Srivastava 等人提出了能够以几乎任意深度优化网络的高速网络。其网络的输出是:

$$x_{l+1} = \phi_{l+1}(x_l, W_H) \cdot \tau_{l+1}(x_l, W_T) + x_l \cdot (1 - \tau(x_l, W_T)) \tag{36}$$

其中 x_l 和 x_{l+1} 对应第 l^{th} 块的输入和输出，$\tau(\cdot)$ 是变换门，$\phi(\cdot)$ 通常是一个仿射变换后的非线性激活函数 (通常它也可能采取其他形式)。这种门机制迫使该层的输入和输出具有相同的大小，并允许数十或数百层的高速公路网络进行有效训练。不同的输入样例，门的输出有很大的不同，说明网络不仅学习固定的结构，而且会根据特定的样例动态路由数据。

残差网络 (ResNets) 与 LSTM 单元中工作的核心思想相同。ResNets 中的快捷连接没有门控，未转换的输入直接传播到输出，从而带来更少的参数，而不是为神经元特定的门控使用可学习的权值。ResNets 的输出如下所示:

$$x_{l+1} = x_l + f_{l+1}(x_l, W_F) \tag{37}$$

其中 f_l 是权值层，它可以是卷积、BN、ReLU 或池化等操作的复合函数。对于残差块，任何较深单元的激活都可以写成较浅单元激活和残差块函数之和。这也意味着梯度可以直接传播到较浅的单位，这使得深度 ResNets 比原来的映射函数更容易优化，也更有效地训练非常深的网。这与通常的前馈网络形成了对比，后者的梯度本质上是一系列矩阵-向量乘积，随着网络的深入，它们可能会消失。

在最初的 ResNets 之后，He 等人 [123] 继续使用另一种 ResNets 预激活变体，在那里他们进行了一系列实验，以证明快捷连接是网络最容易学习的。他们还发现，将 BN 前置比在加法后使用 BN 性能要好得多。在他们的比较中，使用 BN + ReLU 预激活的残差网比之前的 ResNets 得到更高的精度。Shen 等在卷积层的输出中引入一个加权因子，逐步引入可训练层。最新的 Inception-v4 论文还报告了通过跨 Inception 模块使用跳跃连接来加速训练和提高性能。原来的 ResNets 和预激活 ResNets 非常深，但也非常薄。而 Wide ResNets 则提出减小深度，增加宽度，在 CIF AR-10、CIF AR-100 和 SVHN 上取得了令人印象深刻的结果。然而，他们的说法并没有在 Imagenet dataset[1] 上的大规模图像分类任务中得到验证。随机深度 ResNets 随机丢弃层的子集，并通过每个小批量的映射绕过它们。Singh 等人将随机深度 ResNets 和 Dropout 结合起来，推广了 Dropout 和具有随机深度的网络，可以将其视为 ResNets、Dropout ResNets 和随机深度

[1] http://www.image-net.org

ResNets 的集合。ResNets (RiR) 论文中的 ResNets 描述了一种融合经典卷积网络和残差网络的架构,其中 RiR 的每个块包含残差单元和非残差块。RiR 可以知道每个残差块应该使用多少个卷积层。ResNets of ResNets(RoR) 是对 ResNets 架构的修改，该架构建议使用多级快捷连接，而不是先前在 ResNets 上的工作中的单层快捷连接。DenseNet 可以看作是一种将跳跃连接的洞察力发挥到极致的架构，其中一层的输出连接到该模块中所有后续层。在所有的 ResNets 中，Highway 和 Inception 网络中，我们可以看到一个相当明显的趋势，即使用捷径连接来帮助训练非常深的网络。

4. CNNs 的快速处理

随着计算机视觉和机器学习任务的不断挑战，深度神经网络的模型变得越来越复杂。这些强大的模型需要更多的数据进行训练，以避免过拟合。同时，大的训练数据也带来了新的挑战，如怎样在可行的时间内训练网络。在这一节中，我们将介绍一些 CNNs 的快速处理方法。

4.1. FFT

Mathieu 等人利用 FFTs 在傅里叶域进行卷积运算。使用基于 FFT 的方法有许多优点。首先，滤波器的傅里叶变换可以重复使用，因为滤波器与多幅图像在一个小批量进行卷积。其次，输出梯度的傅里叶变换可以在向滤波器和输入图像反向传播梯度时重用。最后，对输入通道的求和可以在傅里叶域中进行，这样每个图像的每个输出通道只需要进行一次傅里叶逆变换。已经开发了一些基于 gpu 的库来加快训练和测试过程，比如 cuDNN 和 fbfft。然而，使用 FFT 进行卷积需要额外的内存来存储傅里叶域的特征映射，因为滤波器必须被填充成与输入相同的大小。当步长参数大于 1 时，这种代价尤其昂贵，这在许多先进的网络中很常见，如一些论文中的早期层。FFT 可以实现更快的训练和测试过程，而小尺寸卷积滤波器的日益突出已经成为 CNNs(如 ResNet 和 GoogleNet) 的重要组成部分,这使得一种专门针对小尺寸滤波器的新方法:Winograd 的最小滤波算法。Winograd 的想法类似于 FFT，在应用逆变换之前，可以在变换空间中跨通道减少 Winograd 卷积，从而使推理更加有效。

4.2. 结构转换

低秩矩阵分解在各种情况下被用来改进优化问题。给定一个秩为 r 的 m×n 矩阵 C，存在一个分解因式 $C = AB$,其中 A 是 m×r 满列秩矩阵,B 是 r×n 满行秩矩阵。因此，我们可以用 A 和 B 来代替 C。要将 C 的参数减少一个分数 p，必须保证 $mr+rn < pmn$，即 C 的秩应满足 $r < pmn/(m+n)$。通过这种分解，空间复杂度从 $O(mn)$ 降低到 $O(r(m+n))$，时间复杂度从 $O(mn)$ 降低到 $O(r(m+n))$。为此，Sainath 等人将低秩矩阵分解应用于深度 CNN 的最终权重层，在精度损失较小的情况下，提高了约 30-50% 的训练速度。同样，Xue 等人在深度 CNN 的每一层上应用奇异值分解，使模型规模减少 71%，相对精度损失小于 1%。Denton 等人和 Jderberg 等人受深度神经网络参数冗余的启发，独立研究了卷积滤波器中的冗余，并发展了近似来减少所需的计算。Novikov 等人推广了低秩的思想，他们将权矩阵视为多维张量，并应用 Tensor-Train 分解来减少全连接层的参数数量。

参考文献

[1] J. Gu, Z. Wang, J. Kuen, L. Ma, and W. Gang. Recent advances in convolutional neural networks. Pattern Recognition, 2015.