# 必须知道深层神经网络中的提示/技巧

<http://lamda.nju.edu.cn/weixs/project/CNNTricks/CNNTricks.html>

## 前言

深度神经网络，尤其是***卷积神经网络***（*CNN*）允许由多个处理层组成的计算模型，以学习多个抽象层次的数据表示。这些方法极大地改进了视觉对象识别，对象检测，文本识别和许多其他领域（如药物发现和基因组学）的现状。

另外，在这个主题中已经发表了许多实用的论文，并且已经提供了一些高质量的开源CNN软件包。还有精心编写的CNN教程或CNN软件手册。然而，它可能缺乏关于如何从头开始实现优秀的深度卷积神经网络的细节的最新和全面的总结。因此，我们收集并总结了DCNN的许多实施细节。**在这里，我们将介绍这些广泛的实施细节，即*技巧*或*技巧*，用于构建和培训自己的深度网络。**

## 介绍

我们假设您已经了解了深度学习的基本知识，在这里我们将在深度神经网络中介绍实现细节（技巧或提示），尤其是与图像相关任务的CNN，主要包括**八个方面**：**1）***数据增强***2）***对图像进行预处理***3）***网络的初始化***4）***训练期间的一些技巧***5）***激活函数的选择***6）***各种正规化***7）***从图中发现的一些见解，*以及最后**8）** *集合多个深度网络的方法*。

另外，相应的幻灯片可在[[幻灯片]中找到](http://lamda.nju.edu.cn/weixs/slide/CNNTricks_slide.pdf)。如果在这些材料和幻灯片中有任何问题/错误，或者有什么重要/有趣的事情，您可以考虑加入，请随时与[我](http://lamda.nju.edu.cn/weixs/)联系。

## 第一章：数据增强

由于深层网络需要在大量训练图像上训练以获得令人满意的性能，如果原始图像数据集包含有限的训练图像，则最好进行数据增强以提高性能。另外，数据增强成为训练深度网络时必须要做的事情。

数据增强有许多方法，例如流行的**水平翻转**，**随机裁剪**和**颜色抖动**。此外，您可以尝试多种不同处理的组合，例如，同时进行旋转和随机缩放。此外，您可以尝试将一个图片中的所有像素的饱和度和值（HSV色彩空间的S和V分量）乘以0.7到1.4之间的参数，并增加一个-0.1到0.1之间的值使其提高到0.25至4之间。对于图像/色块中的所有像素的色调(HSV的H分量)之间添加[-0.1，0.1]之间的值.

Krizhevsky在2012年训练著名的Alex-Net时提出了fancy PCA。Fancy PCA 改变了训练图像集中RGB通道的强度。实际上，您可以首先在整个训练图像上对RGB像素值执行PCA。然后对每张训练图片中的每个PGB图像像素值只用加上以下值(i.e.,I_{xy}=[I_{xy}^R,I_{xy}^G,I_{xy}^B]^T):[bf{p}_1,bf{p}_2,bf{p}_3][alpha_1 lambda_1,alpha_2 lambda_2,alpha_3 lambda_3]^T.其中Pi 和I 是3\*3RGB像素值的协方差矩阵的第i个特征值和特征向量。是从均值为0，标准偏差是0,1的高斯分布中提取的随机值。注意对于特定训练图像的所有像素仅被绘制一次，直到该图像再次用于训练。也就是说，当模型再次遇到相同的训练图像时，它将随机产生另一个alpha_i数据增强。他们声称fancy PCA可以近似捕捉自然图像的重要属性，即对象对光照强度和颜色的变化是不变的”。在分类性能方面，该方案在ImageNet 2012的竞争中降低了1％以上的错误率获得了第一名。

## 第二章：数据预处理

现在我们已经获得了大量训练样本，但请不要着急！实际上，有必要对这些图像/作物进行预处理。在本节中，我们将介绍几种预处理方法。

第一种简单的预处理方法是对数据进行零中心处理，然后对它们进行规范化处理，它们以两行Python代码表示，如下所示：

>>> X -= np.mean(X, axis = 0) #0中心化，每个值减去其平均值

>>> X /= np.std(X, axis = 0) # normalize将处理后的数据除以其标准差

其中，X是输入数据（NumIns×NumDim）。此预处理的另一种形式将每个维度标准化，以使沿维度的最小值和最大值分别为-1和1。如果你的输入特征具有不同的比例尺。这个预处理方式十分有意义，和学习算法的选择可以排在相同的地位。但是对于图像的处理，像素的比例尺已经大致相等（范围从0~255）因此不用执行这种预处理操作。

与第一**种类**似的另一种预处理方法是**PCA Whitening**。在这个过程中，数据首先如上所述居中。然后，您可以计算协方差矩阵，它告诉我们关于数据中的相关结构：

>>> X -= np.mean(X, axis = 0) # zero-center

>>> cov = np.dot(X.T, X) / X.shape[0] # 计算协方差矩阵

之后，通过将原始数据（但以零为中心）投影到特征基元中，解除数据关联：

>>> U，S，V = np.linalg.svd（cov）＃计算数据协方差矩阵的SVD因式分解

>>> Xrot = np.dot（X，U）＃解相关数据

最后一个转换是美白，它将特征值中的数据和特征值的每个维度相除，以规模化：

>>> Xwhite = Xrot / np.sqrt（S + 1e-5）＃除以特征值（它们是奇异值的平方根）

请注意，这里增加了1e-5（或者一个小常量）以防止被零除。这种转变的一个缺点是，它会大大夸大数据中的噪音，因为它将所有维度（包括主要是噪音的微小差异的不相关维度）延伸到输入中的大小相等。实际上，这可以通过更强的平滑来减轻（即增加1e-5成为更大的数字）。

请注意，我们在这里描述这些预处理只是为了完整性。实际上，这些转换不用于卷积神经网络。但是，对数据进行**零中心**也很重要，并且每个像素的**标准化**也很常见。

## 第三章：初始化

现在数据已准备就绪。但是，在开始训练网络之前，您必须初始化其参数。

### 全零初始化

在理想的情况下，通过正确的数据规范化，假设大约一半的权重是正数而其中一半是负数是合理的。一个合理的观点可能是将*所有的初始权重设置为零*，你期望它是期望中的“最佳猜测”。但是，事实证明这是一个错误，因为如果网络中的每个神经元都计算出相同的输出，那么它们也将在反向传播期间计算相同的梯度并进行完全相同的参数更新。换句话说，如果它们的权重被初始化为相同，那么神经元之间就不存在不对称的来源。

### 用小随机数进行初始化

因此，您仍然希望权重非常接近零，但不是相同的零。这样，您可以将这些神经元随机分配到非常接近零的小数字，并将其视为*对称性破坏*。这个想法是，神经元在开始时都是随机的和独特的，所以他们将计算独特的更新并将自己整合为完整网络的不同部分。权重的实现可能简单地看起来像weightssim 0.001times N（0,1），其中N（0,1）是零均值单位标准偏差高斯分布。也可以使用统一分布中的小数字，但这似乎对实践中的最终表现影响相对较小。

### 校准差异

上述建议的一个问题是来自随机初始化神经元的输出分布具有随着输入数量增长的变化。事实证明，通过缩放它的权重向量的*扇入的*平方根（即输入的数量），可以将每个神经元输出的方差标准化为1，如下所示：

>>> w = np.random.randn（n）/ sqrt（n）＃用1 / sqrt（n）校正方差，

其中“randn”是前述的高斯，“n”是其输入的数目。这确保了网络中的所有神经元最初具有大致相同的输出分布并且经验地提高了收敛速度。详细的派生可以从Page找到。18到23的幻灯片。请注意，在推导中，它不考虑ReLU神经元的影响。

### 当前建议

如前所述，通过校准神经元方差的先前初始化不考虑ReLU。He等人最近关于这个主题的论文。[[4]](http://arxiv.org/abs/1502.01852)专门为ReLU推导初始值，得出网络中神经元方差应该如下的结论2.0 / n的：

>>> w = np.random.randn（n）\* sqrt（2.0 / n）＃当前建议

这是目前在实践中使用的建议，正如[[4]中](http://arxiv.org/abs/1502.01852)所讨论的。

## 第四章：训练建议

现在一切准备就绪，我们开始训练神经网络吧。

### 卷积核和池化层的大小设置

训练期间，输入图像的大小优选为2的幂，诸如32（例如，*CIFAR-10*），*64,224*（例如，常用的*ImageNet*），384或512等。更重要的是，我们应该选用小的卷积核（3\*3）和小的步长（1）以及设置0填充（O padding）这不仅减少了参数的数量，而且提高了整个深度网络的准确率。同时，上面提到的一个特例，即3\*3步长为1的滤镜可以保留图像/特征图的空间大小。对于池化层，常用的池化大小是2\*2。

### 学习率

另外，正如Ilya Sutskever的博客[[2]](http://yyue.blogspot.sg/2015/01/a-brief-overview-of-deep-learning.html/)所描述的那样，他建议按照小批量大小（mini-batch-size）来划分梯度。因此，如果您更改小批量大小，则不应始终更改学习率（LR）。（即如果你更改batch\_size则不应该同时更改learning rate）为了获得合适的LR，利用验证集是一种有效的方法。通常，训练开始时LR的典型值是0.1。在实践中，如果你看到你停止在验证集上取得进展，将LR除以2（或5），并继续前进，这可能会给你一个惊喜。

### 对预先训练的模型进行微调。

如今，许多着名的研究小组即[*Caffe Model Zoo*](https://github.com/BVLC/caffe/wiki/Model-Zoo)和[*VGG Group*](http://www.vlfeat.org/matconvnet/pretrained/)发布了许多先进的深度网络。借助预先训练好的深层模型的精彩泛化能力，您可以直接将这些预先训练好的模型用于您自己的应用程序。为了进一步提高数据集的分类性能，一种非常简单而有效的方法是根据您自己的数据对预先训练好的模型进行微调。如下表所示，两个最重要的因素是新数据集的大小（小或大）以及它与原始数据集的相似程度。不同的情况下可以使用不同的微调策略。例如，一个好例子就是您的新数据集与用于培训预先训练的模型的数据非常相似。在这种情况下，如果您的数据非常少，则只需对从预训练模型顶层提取的特征训练一个线性分类器即可。如果您手边有相当多的数据，请用小的学习率对一些预训练模型的顶层进行微调。但是，如果您自己的数据集与预先训练的模型中使用的数据完全不同，但是具有足够的训练图像，则应该对数据进行大量图层调整，并且只需少量学习率即可提高性能。但是，如果您的数据集不仅包含很少的数据，而且与预先训练的模型中使用的数据非常不同，那么您将遇到麻烦。由于数据有限，因此只训练线性分类器似乎更好。由于数据集非常不同，因此从网络顶部训练分类器可能不是最好的，它包含更多的数据集特定功能。

|  |
| --- |
| 表 |

在预先训练的模型上微调您的数据。微调的不同策略被用于不同的情况。对于数据集，Caltech-101与ImageNet相似，其中两个都是以对象为中心的图像数据集;而Place数据库与ImageNet不同，其中一个是以场景为中心的，另一个是以对象为中心的。

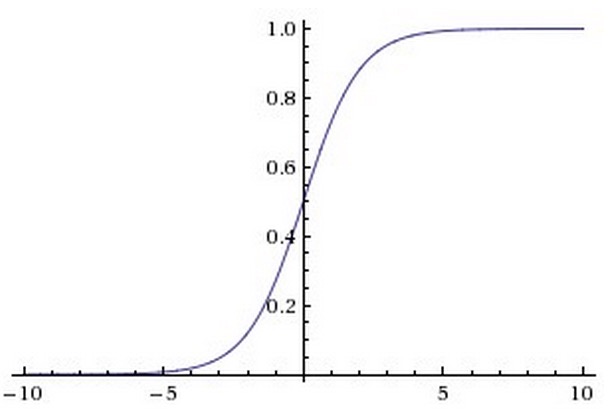
## 第五章：激活函数

深度网络中的关键因素之一是*激活函数*，它将**非线性**引入网络。这里我们将介绍一些流行的激活函数的细节和特性，并在本节后面给出建议。

|  |
| --- |
| 神经元 |

### Sigmoid

Sigmoid函数S型非线性具有数学形式西格玛（X）= 1 /（1 + E ^ { -  X}）。它需要一个实数值并将其“压”到0到1之间的范围内。特别是，大的负数变为0并且大的正数变为1.Shmoid函数历史上经常使用，因为它具有很好的解释，好比神经元的放电，小的放电会使输出成为0，而饱和状态下的放电是1。



实际上，S形非线性最近已经不受欢迎，并且很少被使用。它有两个主要缺点：

饱和状态下的神经元会使梯度消失：S形神经元非常不受欢迎的特性是，当神经元的激活在0或1的尾部饱和时，这些区域的梯度几乎为零。回想一下，在反向传播过程中，这个（局部）梯度将被乘以该门的输出对于整个目标的梯度。因此，如果局部梯度非常小，它将有效地“杀死”梯度，几乎没有信号会通过神经元流向其权重并递归到其数据。此外，在初始化S形神经元的重量以防止饱和时，必须格外小心。例如，如果初始权重太大，那么大多数神经元会变得饱和，网络几乎不会学习。

Sigmoid输出不是以零为中心的。这是不可取的，因为在神经网络的稍后处理层中的神经元（稍后更多）将接收不是以零为中心的数据。这对梯度下降过程中的动力学有影响，因为如果进入神经元的数据总是正的，那么权重上的梯度将在反向传播期间变为全部为正或全部为负（取决于整个表达式的梯度F）。这可能在权重的梯度更新中引入不希望的锯齿形动态。但是，请注意，一旦这些梯度被累加到一批数据中，权重的最终更新可能会有可变的符号，这有点缓解了这个问题。因此，这是一个不便之处，但与上面的饱和激活问题相比，其后果不那么严重。

### RELU

整流线性单元（ReLU）在过去几年中变得非常流行。它计算函数F（X）= MAX（0，x）的，它简单地被限制在零。

使用ReLUs有几个优点和缺点：

（优点）与涉及昂贵操作（指数等）的sigmoid / tanh神经元相比，ReLU可以通过将激活矩阵简单地设置为零来实现。同时，ReLUs不会饱和。

（Pros）与S形/ tanh函数相比，发现随机梯度下降的收敛速度显着加快（例如[1]中的一个因子6 ）。有人认为，这是由于其线性非饱和形式。

（缺点）不幸的是，在训练过程中，ReLU单位可能会变得脆弱，并可能“死亡”。例如，流经ReLU神经元的大梯度可能导致权重更新，使得神经元不会再次在任何数据点上激活。如果发生这种情况，那么从该点开始流经该单元的梯度将永远为零。也就是说，ReLU单位在训练过程中可能会不可逆转地死亡，因为它们可能会从数据流形中剔除。例如，如果学习率设置得太高，您可能会发现多达40％的网络可能“死亡”（即从未在整个训练数据集中激活的神经元）。通过适当设置学习率，这个问题就不那么常见了。

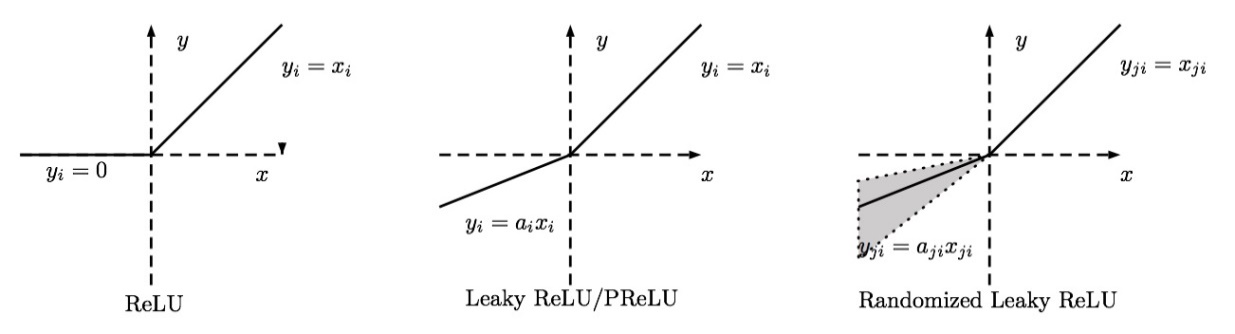
### Leaky ReLU

泄漏的ReLUs是解决“垂死的ReLU”问题的一种尝试。代替函数为零时x <0的，泄漏的ReLU将具有小的负斜率（大约为0.01左右）。也就是说，函数计算f（x）=αx if x <0和f（x）= X if x>=0，并且α是一个小常量。有些人用这种形式的激活函数报告成功，但结果并不总是一致的。

### Parametric ReLU

如今，提出了更广泛的激活职能类别，在下面，我们将讨论ReLU的变种。

**ReLU，Leaky ReLU，PReLU和RRELU**。在这些数字中，对于PRULU而言，alpha_i是学习得到的和对于Leaky ReLU而言 alpha_i是固定的。对于RReLU，alpha_ {}纪是一个随机变量在给定范围内保持采样，并在测试中保持不变。

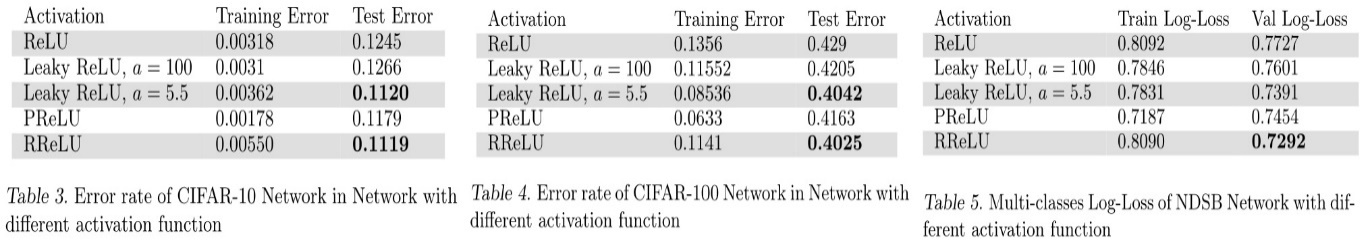


第一个变体称为参数整流线性单元（PReLU）[[4]](http://arxiv.org/abs/1502.01852)。在PReLU中，负数部分的斜率是从数据中学习而不是预先定义的。他等人。[[4]](http://arxiv.org/abs/1502.01852)声称PReLU是[ImageNet](http://www.image-net.org/)分类任务中超越人类表现的关键因素。PReLU的反向传播和更新过程非常简单，类似于传统的ReLU，对应PPT中43页对其有详细的介绍。

### Randomized ReLU

第二个变体称为随机整流线性单元（RReLU）。在RRELU中，负面部分的斜率在训练中的给定范围内随机化，然后在测试中固定。正如[[5]中](http://arxiv.org/abs/1505.00853)提到的，在最近的Kaggle[国家数据科学博物馆（NDSB）](https://www.kaggle.com/c/datasciencebowl)竞赛中，据报道，由于RRELU的随机性，RRELU可以减少过度拟合。此外，由NDSB竞赛获胜者提出。在训练过程中是从1/U（3,8）的分布中随机取值的，而在test数据集中固定在其期望的数值即2/（l+u）=2/11。

在[[5]中](http://arxiv.org/abs/1505.00853)，作者评估了*CIFAR-10*，*CIFAR-100*和*NDSB*数据集上具有不同激活函数的*两种现代* CNN架构的分类性能，如下表所示。*请注意，对于这两个网络来说，每个卷积层都有一个激活函数。这里的实际上表示的是ReLU公式中的1/*



从这些表格中，我们可以发现ReLU的性能不是所有三个数据集的最佳表现。对于Leaky ReLU，较大的斜率α将获得更好的准确率。PReLU很容易适用于小数据集（其训练错误最小，而测试错误不令人满意），但仍然优于ReLU。此外，RReLU明显优于NDSB上的其他激活功能，这表明RReLU可以克服过度拟合，因为该数据集的训练数据少于CIFAR-10 / CIFAR-100。***总之，在这三个数据集中，三种类型的ReLU变体都始终优于原始ReLU。PReLU和RRELU似乎是更好的选择。此外，何*等人*。也报道了类似的结论***[***[4]***](http://arxiv.org/abs/1502.01852)。

## 第六章：正则化

有几种方法可以控制神经网络防止过拟合的能力：

### L2正规化

L2正规化也许是正规化最常见的形式。它可以通过直接在目标中惩罚所有参数的平方幅度来实现。也就是说，对于w ^网络中的每个权重，我们将该术语添加frac {1} {2} lambda w ^ 2到目标中，拉姆达正则化强度在哪里。通常会看到压裂{1} {2}前面的因子，因为这个关于参数的这个项的梯度w ^是简单的拉姆达瓦特而不是2Lambda w。L2规则化具有严重惩罚峰值权重向量并优选漫反射权重向量的直观解释。

### L1正则化

**L1正则化**是另一种相对常见的正规化形式，其中对于每个权重，w ^我们将该术语添加lambda | w |到目标中。将L1正则化与L2正则化相结合是可能的lambda_1 | w | + lambda_2 w ^ 2（这称为[弹性净正则化）](http://web.stanford.edu/~hastie/Papers/B67.2%20%282005%29%20301-320%20Zou%20&%20Hastie.pdf)）。L1正则化具有令人感兴趣的性质，它使得权重向量在优化过程中变得稀疏（即非常接近零）。换句话说，具有L1正则化的神经元最终只使用其最重要输入的稀疏子集，并且变得对“噪声”输入几乎不变。相比之下，来自L2正则化的最终权重向量通常是弥散的，小数字。在实践中，如果你不关心显式特征选择，那么L2正则化可以预期比L1更好的性能。

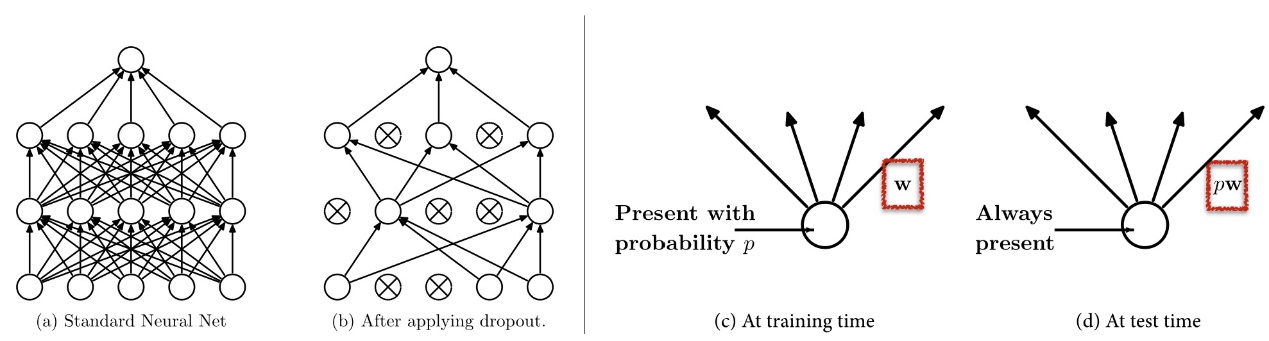
### 最大范数约束。

最大范数约束。正则化的另一种形式是对每个神经元的权向量的强度执行绝对上限，并使用投影梯度下降来强制约束。在实践中，这对应于正常执行参数更新，然后通过钳制VEC【W】每个神经元的权向量来满足约束来执行约束平行vec {w} parallel_2 <c。典型值C大约为3或4.有些人在使用这种形式的正则化时报告了改进。其吸引人的特性之一是，即使学习率设置得太高，网络也不能“爆炸”，因为更新总是有界的。

### Dropout

**Dropout**是Srivastava *等人*非常有效，简单且最近引入的正则化技术。在[[6]](http://jmlr.org/papers/v15/srivastava14a.html)中补充了其他方法（L1，L2，maxnorm）。在训练期间，丢失可以被解释为在完整的神经网络中对神经网络进行采样，并且仅基于输入数据更新采样网络的参数。（然而，可能的采样网络的指数数量并不是独立的，因为它们共享参数）。在测试期间没有应用丢弃，并且解释评估所有子网络的指数大小的整体上的平均预测（更多关于在下一节合奏）。在实践中，Dropout率的值P = 0.5 是合理的默认值，但可以根据验证数据进行调整。

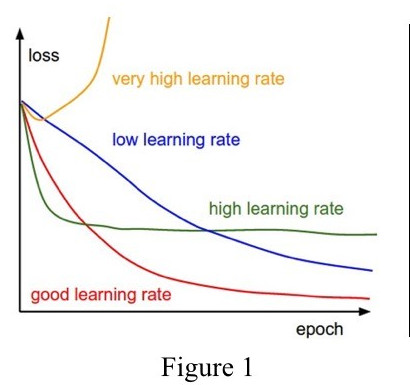
最常用的正则化技术Dropout [[6]](http://jmlr.org/papers/v15/srivastava14a.html)。训练时，退出是通过仅保持神经元以某种概率p（超参数）为活动状态来实现的，否则将其设置为零。此外，谷歌为Dropout于2014年申请了[美国专利](https://www.google.com/patents/WO2014105866A1)。



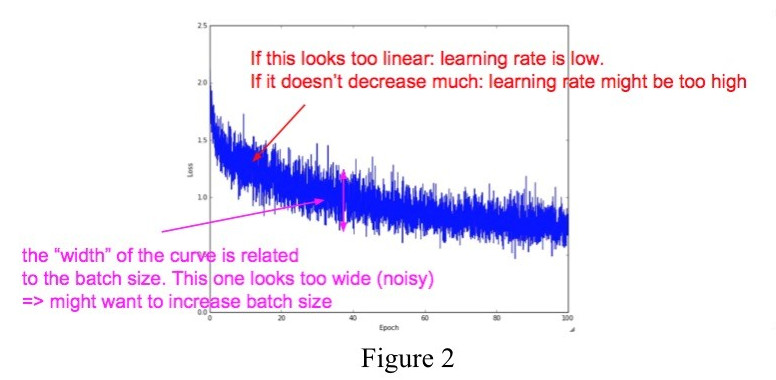
## 第七章：使用图像调节参数

最后，从上面的提示中，您可以获得满意的设置（例如，数据处理，体系结构选择和细节等），以适应您自己的深层网络。在训练期间，您可以画出一些图像来表明您的网络的训练效果。

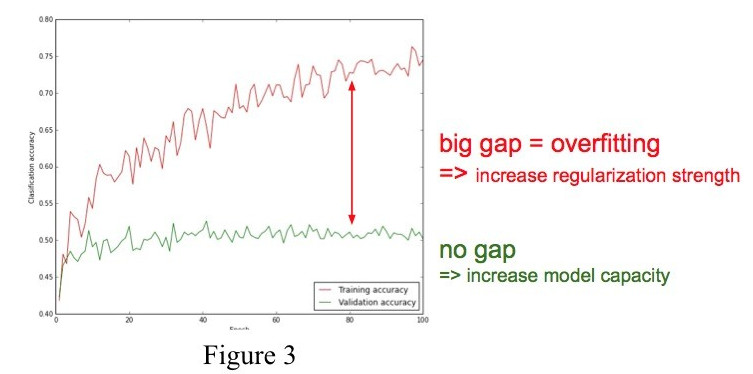
**如我们所知，学习率非常敏感。从下面的图1中可以看出，很高的学习率会导致一个非常奇怪的损失曲线。即使经过大量的时代，低学习率也会让你的训练损失非常缓慢地下降。相反，高学习率会使训练损失在开始时迅速下降，但也会降至当地最低水平。因此，在这种情况下，您的网络可能无法取得令人满意的结果。对于良好的学习速度，如图1所示的红线，其损失曲线平稳运行，最终达到最佳性能。**

****

**现在让我们放大损失曲线。时代呈现训练数据一次训练的次数，因此每个时代都有多个迷你批次。如果我们绘制每个培训批次的分类损失，则曲线如图2所示。与图1类似，如果损失曲线的趋势看起来过于线性，则表示您的学习率较低; 如果它减少不了多少，它会告诉你学习率可能太高。此外，曲线的“宽度”与批量大小有关。如果“宽度”看起来太宽，也就是说每个批次之间的差异过大，这指出应该增加批量大小。**

****

**另一个提示来自精度曲线。如图3所示，红线是训练的准确性，绿线是验证的线。当验证准确性收敛时，红线和绿线之间的差距将显示您的深层网络的有效性。如果差距很大，则表明您的网络可以在训练数据上获得较高的准确性，而在验证集上它只能达到较低的准确性。很显然，你的深层模型适合训练集。因此，你应该增加深层网络的规则化强度。然而，低精度水平的差距并不是一件好事，这表明你的深层模型的可学性低。在这种情况下，最好增加模型容量以获得更好的结果。**

****

## 第八章：集成方法

在机器学习中，集成方法[[8]](https://www.crcpress.com/Ensemble-Methods-Foundations-and-Algorithms/Zhou/9781439830031)培养多个学习者，然后将它们结合起来使用，这是一种最先进的学习方法。众所周知，合奏通常比单个学习者精确得多，并且合奏方法在许多实际任务中已经取得了巨大成功。在实际应用中，尤其是挑战或竞赛中，几乎所有的第一名和第二名获胜者都使用集成方法。

### 相同的模型不同的初始化方法

使用交叉验证来决定最好的超参数，然后使用这些超参数但是使用不同的初始化方式来训练多个模型。这种方法的带来的区别仅仅是由于参数初始化带来的。

### 交叉验证期间发现的顶级模型

使用交叉验证来确定最佳的超参数，然后选择前几个（例如10个）模型来形成整体。这改善了整体的多样性，但具有包含次优模型的危险。在实践中，这可以更容易执行，因为它不需要交叉验证后对模型进行额外的再训练。实际上，您可以直接从[Caffe Model Zoo中](https://github.com/BVLC/caffe/wiki/Model-Zoo)选择几种最先进的深层模型来执行合奏。

### 对于同一模型设置不同的检查点

如果训练十分的昂贵，一些人在一个模型（例如在每一轮迭代之后）设立的不同检查点没有取得好的效果，然后使用当前的网络进行集成学习，很明显我们不可能得到非常好的结果，但是实际上也工作得效果不错，这种方法的优点就是十分的便宜。

### 一些实际的例子

如果你的视觉任务与高级图像语义相关，例如来自静止图像的事件识别，则更好的集合方法是采用在不同数据源上训练的多个深度模型来提取不同的和互补的深度表示。例如，在与[ICCV'15](http://pamitc.org/iccv15/)相关的[文化活动识别](https://www.codalab.org/competitions/4081#learn_the_details)挑战中，我们利用了五个不同的深度模型，对[ImageNet的](http://www.image-net.org/)图像，[场所数据库](http://places.csail.mit.edu/)和由[比赛组织者](http://gesture.chalearn.org/)提供的文化图像进行了训练。之后，我们提取了五个互补的深层特征，并将它们视为多视图数据。结合[[7]中](http://lamda.nju.edu.cn/weixs/publication/iccvw15_CER.pdf)描述的“早期融合”和“后期融合”策略，，我们取得了最好的成绩之一，并在这一挑战中排名第二。与我们的工作类似，[[9]](http://cs231n.stanford.edu/reports/milad_final_report.pdf)提出了*Stacked NN*框架来同时融合更深层的网络。

## 第九章：杂项

在现实世界的应用程序中，数据通常是不平衡的类别：一些类别具有大量的图像/训练实例，而一些具有非常有限的图像数量。正如最近的一份技术报告[10]中所讨论的那样，当深度CNN在这些不平衡的训练集上进行训练时，结果表明，不平衡的训练数据可能会对深度网络的整体性能产生严重的负面影响。对于这个问题，最简单的方法是通过对不平衡数据直接上采样和下采样来平衡训练数据，如[10]所示。另一个有趣的解决方案是我们在这具有挑战性的解决方案中引入一种裁剪[7]。由于原始文化事件图像不平衡，我们仅仅只从少量训练图像的类中提取数据，一方面可以提供多种数据源，另一方面可以解决类不平衡问题。此外，您可以调整克服类不平衡的微调策略。例如，您可以将自己的数据集分为两部分：一部分包含具有大量训练样本的类;另一个包含有限数量样本的类别。在每个部分，阶级失衡问题不会很严重。在对数据集进行微调开始时，首先对具有大量训练样本的类进行微调，然后再对具有有限样本的类别进行微调。

## References & Source Links

1. A. Krizhevsky, I. Sutskever, and G. E. Hinton. [ImageNet Classification with Deep Convolutional Neural Networks](http://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf). In *NIPS*, 2012
2. [A Brief Overview of Deep Learning](http://yyue.blogspot.com/2015/01/a-brief-overview-of-deep-learning.html/), which is a guest post by [Ilya Sutskever](http://www.cs.toronto.edu/~ilya/).
3. [CS231n: Convolutional Neural Networks for Visual Recognition](http://cs231n.stanford.edu/index.html/) of *Stanford University*, held by [Prof. Fei-Fei Li](http://vision.stanford.edu/index.html) and [Andrej Karpathy](http://cs.stanford.edu/people/karpathy/).
4. K. He, X. Zhang, S. Ren, and J. Sun. [Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification](http://arxiv.org/abs/1502.01852). In *ICCV*, 2015.
5. B. Xu, N. Wang, T. Chen, and M. Li. [Empirical Evaluation of Rectified Activations in Convolution Network](http://arxiv.org/abs/1505.00853). In *ICML Deep Learning Workshop*, 2015.
6. N. Srivastava, G. E. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. [Dropout: A Simple Way to Prevent Neural Networks from Overfitting](http://jmlr.org/papers/v15/srivastava14a.html). *JMLR*, 15(Jun):1929−1958, 2014.
7. X.-S. Wei, B.-B. Gao, and J. Wu. [Deep Spatial Pyramid Ensemble for Cultural Event Recognition](http://lamda.nju.edu.cn/weixs/publication/iccvw15_CER.pdf). In *ICCV ChaLearn Looking at People Workshop*, 2015.
8. Z.-H. Zhou. [Ensemble Methods: Foundations and Algorithms](https://www.crcpress.com/Ensemble-Methods-Foundations-and-Algorithms/Zhou/9781439830031). *Boca Raton, FL: Chapman & Hall*CRC/, 2012. (ISBN 978-1-439-830031)
9. M. Mohammadi, and S. Das. [S-NN: Stacked Neural Networks](http://cs231n.stanford.edu/reports/milad_final_report.pdf). Project in [*Stanford CS231n Winter Quarter*](http://cs231n.stanford.edu/reports.html), 2015.
10. P. Hensman, and D. Masko. [The Impact of Imbalanced Training Data for Convolutional Neural Networks](http://www.diva-portal.org/smash/get/diva2:811111/FULLTEXT01.pdf). *Degree Project in Computer Science*, DD143X, 2015.