进展报告 ——从感知器到深度学习

霍超凡

2019 年 9 月 1 日

1 概述

前两个月老师让我了解resNet并熟悉深度学习相关工具，经过两个月间断式学习，我了解从上世纪中期到如今为期七十年左右神经网络发展的历程，从单层感知器逐步深入到resNet，并接触深度学习相关基础理论。具体总结如下。

从七月初布置任务开始的第一个星期，我开始了解resNet和相关神经网络的基础理论。七月的第二个星期参加一个夏令营形式的活动，之前的学习暂告一段落。从七月的后两个星期到八月上旬的将近一个月的时间，在青岛校区参加暑期课程，期间进一步了解神经网络相关理论，并逐渐接触深度学习，除此之外，数学建模课程也对自己神经网络的学习有帮助，比如优化理论。八月上旬课程结束，数学建模进入实训阶段，获得了较多的时间自习，主要了解到粒子群智能优化算法和没能来得及学习的禁忌搜索。八月的第三个星期，回到济南并继续进行之前被搁置的神经网络理论的学习，八月后旬，我往返家里和学校，开始接触到深度学习的相关工具，并将之前学到的理论转换为实际。

在第二节，我按照神经网络发展的路线简要串联我了解到的神经网络基础理论，第三节集中讨论关于resNet 相关内容。

2 神经网络基础理论

2.1 模型的建立

MP神经网络模型 1943年，Mcculloch和Pitts提出MP模型[5]，这个模型将单个神经元视为一个输入输出非线性单元，神经元接受的刺激达到一定阈值后才会引起兴奋，并向下一个神经元传递兴奋，神经元之间通过突触传递兴奋，兴奋从前一个神经元的神经末梢到后一个神经元的胞体单向传递。神经元之间的连接强度即突触传递兴奋的能力存在差异，刺激引发神经元兴奋的阈值也随神经元内部性质而相异。神经元N_i和神经元N_j间突触的连接强度用一个数值w_{ij}衡量，w_{ij}与突触自身性质有关。从神经元N_i到神经元N_j的兴奋强度为x_{ij}，兴奋经过突触后，或被增强或被抑制，N_j实际接收到N_i的兴奋强度为y_{ij} 且y_{ij}=w_{ij}·x_{ij}。设与神经元N_j直接相连的神经元个数为m，那么神经元接收到的总兴奋为

$$\widehat{y}_j = \sum_{i=1}^{m} y_{ij} = \sum_{i=1}^{m} w_{ij} x_{ij} \tag{1}$$

在MP模型中，兴奋“全或无”，兴奋强度x_{ij}取逻辑值，即$x_{ij} \in \{0, 1\}$。神经元N_j 传递兴奋的阈值为θ_j，神经元N_j实际输出为

$$y_i = hardlim(\widehat{y}_j - \theta_j) \tag{2}$$

其中$hardlim(\cdot)$是一个非线性函数，当$\widehat{y}_j$大于或等于θ_j时$hardlim$ 输出1，反之输出0。

MP模型使用“全或无”的思想，只考虑到兴奋的有无，却没有对兴奋的强度进行刻画；另一方面MP 模型是一个静态模型，认为当前神经元的活动状态和之前的状态无关。这个假设简化了模型，也不影响实际应用，因为在实际应用中通常神经元的输入数据通常是单个的、一次性的。如果输入神经网络的数据是一个连续序列，比如视频序列，MP模型需要进行更正，神经元的活动受到内外部化学环境的影响，之前的状态会影响当前状态，这种影响可视为兴奋的延时。动态MP模型下，神经元内部各个参数并不是一成不变的，权值w_{ij}应按照如下更新

$$w_{ij}^{(t+1)} = w_{ij}^{(t)} + f(x_{ij}^{(t)}) \tag{3}$$

上式最后一项描述前一状态对当前状态的影响。阈值θ的更新与之类似。关于$f(x)$函数的具体形式需要对神经元传导兴奋的原理的深入理解才能确认，可以根据经验猜测前一状态对当前状态其促进作用，取简单线性函数

$$f(x) = kx(k > 0) \tag{4}$$

最后我们应该批判的看待MP模型，MP模型会随着我们对神经元实际活动机理的深入研究而不断改进。

2.2 学习规则

Hebb学习规则 1949年，Donald Olding Hebb在《The Organization of Behavior》[3]中对提出来一种调整权值的方法，称为Hebb学习规则。Hebb从狗听铃声分泌唾液的实验出发，认为我们学习过程实质上是神经元间突触的建立与松弛，相邻的神经元同时兴奋会导致突触的建立与强化，长期不被使用的突触传递兴奋的能力会减弱。这种思想是我们训练神经网络的基础，无论是δ 规则还是BP反向传播都可以在某种程度看出Hebb规则的影子。

感知器 1958年，Frank Rosenblatt提出单层神经网络Perceptron。感知器只包含一层神经网络，输入和权值点乘后加上一个偏置，最后经过hardlim激活函数。其数学模型如下

$$y = hardlim(w^T x + b) \tag{5}$$

感知器可用于一般的二分类问题，超平面$w^T x + b = 0$将$\mathbb{R}^n$分成两部分，位于超平面一侧的特征点代入方程后得到$w^T x + b > 0$，经过激活函数后得到的输出为1，另一侧得到的输出为0，于是这个网络将特征点分解成两类。关于感知机的学习，存在几种算法，这几种算法出发点不相同，但是殊途同归，得到的结果具有相似的形式。《NND》[2]一书曾经用几何的方式讨论了二维形式的感知机学习规则，将特征点看成n维空间的一个向量，单位权值向量w与分离超平面垂直，当输入向量x在w的投影大于$-b$时，输出为1，反之输出为0。记误分的样本点(x_1, y)，x_1是特征向量，y取值为1 说明为正样本点，取值为0则为负样本点，将x_1代入感知器后得到的输出为$\widehat{y}$，误差$e = y - \widehat{y}$，权值w和b 按照如下原则进行更新

$$\begin{aligned} w_{new} &= w_{old} + ex \\ b_{new} &= b_{old} + e \end{aligned} \tag{6}$$

对于误分的正样本点e取值为1，正样本被无法成负样本，说明该正样本点在权值向量上的投影较小，将权值向量加上x，使权值向量的方向偏向正样本点的方向，同时应该减少门限值，b 会增1。按照Hebb学习规则，正样本经过感知器后输出为1，神经元被激活，相关的突触应该得到强化，对应的权值应该增加。于是神经元的权值应该表达为各个正样本输入特征之和即

$$w = \sum_{i=1}^{m} x_i^+ \tag{7}$$

相反，对于负样本，神经元得到抑制，权值应该得到消减，消减的程度与输入的值有关，于是权值的计算公式为

$$w = \sum_{i=1}^{m_1} x_i^+ - \sum_{j=1}^{m_2} x_j^- \tag{8}$$

最小均方误差 1960年，Bernard Widrow和Ted Hoff提出Adaline网络和最小均方误差的训练方法。Adaline 使用的激活函数与感知器所使用的激活函数不同，激活函数是恒等映射函数$f(x) = x$。最小均方误差LMS 方法是一种性能学习方法，使用输出值与目标值的差异来衡量神经网络的性能，训练网络的过程是提高网络性能的过程。设输入m个样本点$x_1, x_2, x_3, \cdots, x_m$，其中$x \in \mathbb{R}^{n+1}$, 最后一维添加常量1以简化形式。这n个样本点写成矩阵形式，得到$m \times n$ 矩阵$X = (x_1, x_2, \cdots, x_m)^T$，样本点对应的目标输出为$y = (y_1, y_2, \cdots, y_m)^T$，实际输出为$\widehat{y}$。使用目标输出与实际输出的差距衡量网络性能，

$$p = \|y - \widehat{y}\|_2^2 \tag{9}$$

其中$\widehat{y} = Xw$，权值向量w最后一维添加b以简化形式，为提高网络性能，需要最小化实际输出与目标输出，即

$$min\ \|Xw - y\|_2^2 \tag{10}$$

展开化简后得到

$$min\ w^T X^T X w - 2y^T X w + y^T y \tag{11}$$

等价于如下二次规划

$$min\ f(x) = \frac{1}{2} x^T A x - b^T x + c \tag{12}$$

上述二次规划是凸优化问题，只有一个最优点x^*，且最优点处的梯度等于0，即

$$Ax^* - b = 0 \tag{13}$$

如果$A = 2X^T X$可逆，得到x^*的解析表达

$$x^* = A^{-1} b \tag{14}$$

如果输入样本点正交，则有

$$A^{-1} = (2X^T X)^{-1} = 2I \tag{15}$$

于是

$$w^* = A^{-1} b = 4X^T y = 4 \sum_{i=1}^{m} y_i x_i \tag{16}$$

得到和Hebb规则相同的结果。通常样本个数非常庞大，计算A的逆矩阵不现实，而且也无法保证输入样本线性无关。采用分阶段优化的方法或局部搜索方法，一次只输入一部分样本来计算w。这个过程与BP算法不谋而合。

2.3 几何视角下的分类问题

1969年，Marvin Minsky和Seymour Papert 在《Perceptrons: an introduction to computational geometry》[4]书中揭示了单层神经网络只能应用于线性可分的问题，对于诸如异或等线性不可分问题不能解决。此后，神经网络的研究陷入了很长一段时间的低迷期。增加神经网络层数可以解决这个问题，有研究表明，两层的神经网络可以任意精度拟合任意函数。两层的神经网络也可以解决线性不可分的问题。

神经网络技术广泛应用于分类问题，设目前存在待分类的若干类别$T_1, T_2, \cdots, T_k$，类别T_i中的每一个实例用一个特征向量$x = (x_1, x_2, \cdots, x_n)^T$表示，根据特征向量区分不同类别。这些类别的实例所对应的特征向量在高维空间中形成不同点集$S_1, S_2, \cdots, S_k$，把这些点集称为这个类别的区域$D_1, D_2, \cdots, D_k$。假设特征选取的当，同一类别中实例向量在向量空间中形成一个连续的区域，不同类别所形成的区域交集尽可能小，否则，如果不同类别的向量空间高度重合，这说明是特征向量选取不当。

记A为一个$m \times n$矩阵，b为m维向量，$C = \{x \in \mathbb{R}^n | Ax \leqslant b\}$ 表达一个凸区域，任何区域都可以用若干凸区域组合形成，即$D = C_1 \cup C_2 \cup \cdots \cup C_k$。一个分类问题可以如下形式化定义

定义 1 (分类问题). 给定k个类别$T_1, T_2, \cdots, T_k$，类别中任何个体可以用一个n维向量唯一确定，这k个类别存在k个区域$D_1, D_2, \cdots, D_k$与之对应，其中$D_i \subset \mathbb{R}^n$。这些区域连续，且相互没有交集。现随机从k个区域中抽出若干个点形成k个点集$S_{train1}, S_{train2}, \cdots, S_{traink}$。其中$|S_{traini}| = m_i, S_{traini} = \{x_1, x_2, \cdots, x_{m_i}\}, x_j \in \mathbb{R}^n$。这$k$个点集和对应的类别形成一个数据集

$$DB = \{(S_{train1}, T_1), (S_{train2}, T_2), \cdots, (S_{traink}, T_k)\},$$

任务要求根据给定的数据集DB找出一个映射函数

$$\Phi : \mathbb{R}^n \to \{T_1, T_2, \cdots, T_k\},$$

使得该映射函数能够将使用相同的方式从k 个区域中提取若干个点映射到正确的类别。

从几何角度看，分类工作实际上是一一找出这些区域的包络。任何区域都可以看成若干个凸区域的组合，神经网络中第一层感知器表达一个超平面，第二层感知器将多个超平面进行与运算得到一个凸区域，第三层感知器将第二层的凸区域进行或运算形成一般的非凸区域，所以三层神经网络可以表达任意区域。实际上并不需要三层，双层神经网络可以实现任意区域的凸包。第一层多个感知器拟合超平面，第二层感知器将这些超平面连接形成一个区域，只需调整第二层感知器的权值，它可以完成将超平面连接形成凸区域和将凸区域连接形成一般区域的两步逻辑意义不同的操作。经过以上分析，理论上，双层感知器网络可以解决任何分类问题。

神经网络分类的工作原理是找出这些区域的包络，由于从自然界得到的数据并不能够真正的反映这个类别的特征属性，在解决这类问题时两层网络并不能够很好的解决问题，需要我们手动从这些数据中提取特征向量。卷积神经网络是一个提取特征点很好的结构，卷积操作提取局部信息，高层卷积进一步提取信息。一般的用于进行图片分类任务的网络都是由若干卷积层加上最后全连接层组成。

2.4 待续

1974年，Paul Werbos提出BP神经网络算法，该算法普遍应用于神经网络的训练。

RBF网络

Hopfield网络

玻尔兹曼机

网络回归问题，分类问题，解决优化问题。

2006年，Hinton 和他的学生Salakhutdinov发表《Reducing the Dimensionality of Data with Neural Networks》文章，这是深度学习的开端。

深度信念网络（DBN）

卷积神经网络

.....

3 ResNet实践

3.1 ResNet简介

为了追求更强的表现力，我们在对神经网络在深度和广度做出了不同的探索，深层网络最大难题是难以训练，ResNet对训练深层网络做出尝试，并获得很好的结果。作者在训练深层网络时，出现随着网络越深性能下降的现象，理论上一个深层网络的性能应该不比浅层网络差，而实际得到的结果与之相反。这是训练不当的问题，而不是网络结构的问题，当网络越深，使用BP算法反向传播的路径越长，相乘的系数越多，越有可能出现梯度爆炸或梯度消失的问题。

3.2 ResNet中恒等映射的作用

ResNet是由若干残差块堆叠的网络结构，在传统的向前传播卷积网络旁边加上一个恒等映射层，这个恒等映射将输入直接连接到输出，这个‘连接’可以解决梯度消失的问题。采用Li Da在《Classification of remote sensing images based on densely connected convolutional networks》[1]的记号，设一个网络具有L层，每一层可以理解成一个非线性转换$H_\ell(\cdot)$，这里表示网络中的第ℓ层，$H_\ell(\cdot)$ 可以是BN 层、激活层、池化层或卷积层堆叠形成，记x_ℓ为第ℓ 层的输出。那么有

$$x_\ell = H_\ell(x_{\ell-1}) \tag{17}$$

第ℓ层存在m_ℓ个参数，记为$\theta_{\ell,1}, \theta_{\ell,2}, \cdots, \theta_{\ell,m_\ell}$。反向传播时需要计算梯度，而梯度需要计算每一个参数关于输出的偏导数

$$\frac{\partial x_L}{\partial \theta} = \frac{\partial H_L(x_{L-1})}{\partial \theta} = \frac{\partial x_{L-1}}{\partial \theta} \cdot \frac{\partial H_L}{\partial x_{L-1}} \tag{18}$$

那么第ℓ层的参数

$$\frac{\partial x_L}{\partial \theta_\ell} = \frac{\partial H_L}{\partial x_{L-1}} \cdot \frac{\partial H_{L-1}}{\partial x_{L-2}} \cdot \cdots \cdot \frac{\partial x_\ell}{\partial \theta_\ell} \tag{19}$$

对于越靠前面层的参数，需要相乘的偏导越多，所以存在梯度爆炸或梯度消失的风险。

而对于残存网络来说，残差层可以表示为

$$x_\ell = H_\ell(x_{\ell-1}) + x_{\ell-1} \tag{20}$$

上式中，后面对$x_{\ell-1}$求偏导后是1，如果将残差块连续堆叠，第ℓ层层数相对于第L 层输出的偏导为

$$\frac{\partial x_L}{\partial \theta_\ell} = \left(\frac{\partial H_L}{\partial x_{L-1}} \cdot \frac{\partial H_{L-1}}{\partial x_{L-2}} \cdot \cdots \cdot \frac{\partial x_\ell}{\partial \theta_\ell}\right) + \left(\frac{\partial H_{L-1}}{\partial x_{L-2}} \cdot \cdots \cdot \frac{\partial x_\ell}{\partial \theta_\ell}\right) + \cdots + \frac{\partial H_{\ell+1}}{\partial x_\ell} \cdot \frac{\partial x_\ell}{\partial \theta_\ell} + \frac{\partial x_\ell}{\partial \theta_\ell} \tag{21}$$

从上式可以看出，恒等映射的引入使偏导通过恒等映射更快的传到前面。

残差块不仅仅解决了梯度消失的问题，而且在某种程度增加了该层的宽度，恒等映射提高了信息复用率，前面层得到的特征可以和后面层得到的信息综合。所以残差块使网络的深度和宽度都有提升，Li Da[1]基于这样的观察，在残差块的基础上多增加了一些恒等映射路径，相比残差网络信息复用率更高，也获得了更好的性能。

3.3 实验

使用CIFAR10数据集，以Pytorch为开发环境，对ResNet网络进行训练。训练数据为50k的32x32包含10个类别的图片，测试样本容量为10k。使用mini-batch方法对训练样本进行迭代，每迭代若干

次后，记录当前的平均损失、训练精度、测试精度。设记录间隔为k，区间$[iter, iter+k)$内平均损失为

$$loss_{avg} = \frac{1}{k} \cdot \sum_{i=iter}^{i<iter+k} L_i \tag{22}$$

设s_{batch}为一次迭代的训练样本个数，L_i为第i次迭代时的损失值，注意Pytorch计算的损失值是这一个batch内的平均损失，所以这里的分母是k，而不是$k \cdot s_{batch}$。训练精度（后面有时也称作训练正确率）是正确预测的样本数与样本总数的比值，即

$$acc_{train} = \frac{num_{correct}}{k \cdot s_{batch}} \cdot 100 \tag{23}$$

其中$num_{correct}$为当前k次迭代中正确预测的样本总数。

之后的实验围绕平均损失、训练精度、测试精度这三个衡量指标展开一些超参数对训练结果的影响和过拟合现象的解决。

3.4 实验结果

在训练ResNet过程中，一些超参数的设定对于结果有较大的影响。我主要探究了学习率、势能因子和权值衰减率对网络的影响。

学习率 BP算法是训练神经网络的主流方法，BP算法利用优化算法的最速梯度下降方法对权值进行更新，设$L_\theta(x)$是待最小化的损失函数，则

$$\theta^{(i+1)} = \theta^{(i)} - \epsilon \frac{\partial L_\theta(x)}{\partial \theta} \tag{24}$$

在优化理论中，习惯将ϵ称为步长，而在深度学习中ϵ称为学习率。学习率控制参数更新的速度，使用过高的学习率网络具有较快的收敛速度，能够跳过一些局部最小值，但是在训练后期会影响网络的稳定性；使用过小的学习率具有较好稳定性，但是任意陷入局部最优，训练早期也会影响网络的更新速度。

使用CIFAR10数据集，训练深度为34的残差网络，学习率依次为0.1、0.01、0.001、0.0001，得到如图1结果

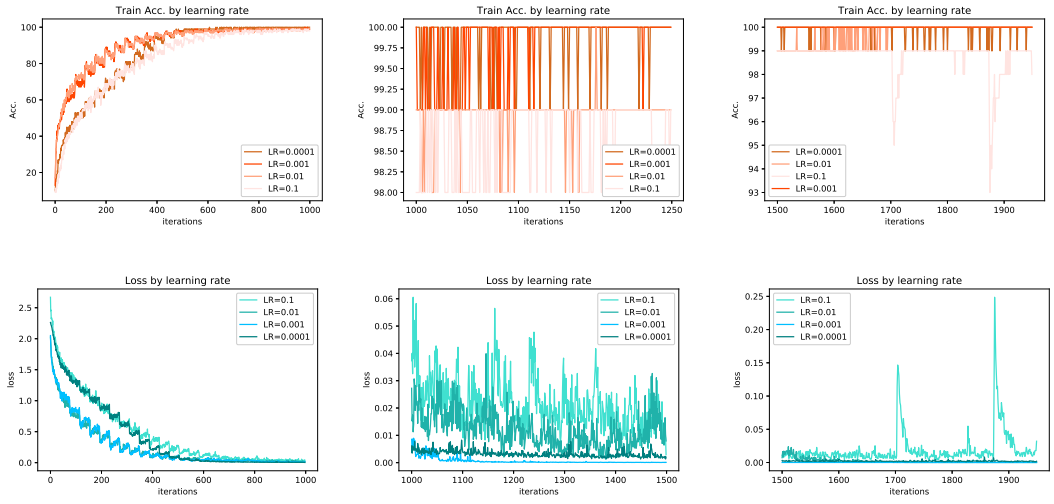

图 1: 不同学习率下准确度和损失与迭代次数关系图

图1分别给出训练不同时期时，loss和训练精度的变化过程。在训练早期，学习速率为0.01 和0.001时，loss具有近乎相同的下降速度，学习速率为0.001明显慢于前两种，学习速率设为0.1 时，下降速度不如0.01和0.001，这可能是由于步长过长跳过了最小值，在最小值的两边局部震荡的缘故。在训练中期，学习速率为0.01和0.001的网络基本收敛，而学习速率稍高的网络仍然在最小值两边震荡。在训练后期，只有学习率为0.1的网络仍然有跳出最小值的趋势，其它三种已经收敛。训练精度跟训练损失相反。

在训练过程中，在测试集的正确率变化如图2。

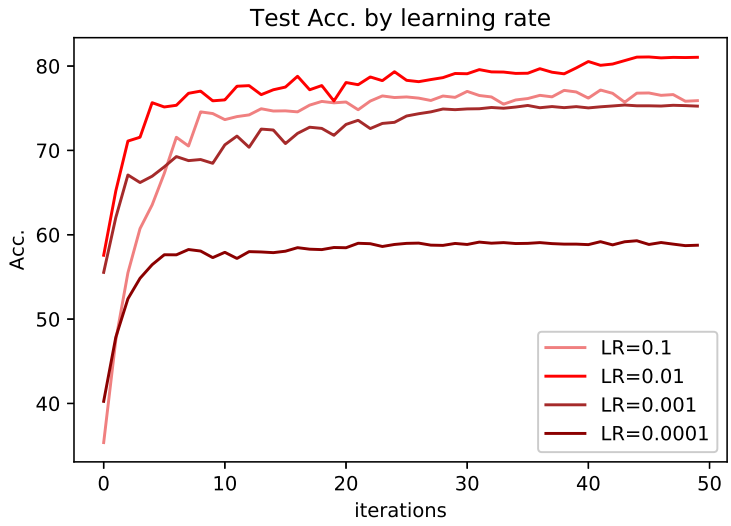

图 2: 不同学习率下测试精度和迭代次数关系图

结果出人意料，虽然在训练集上精度都达到100%，损失达到0，但是在测试集却相差悬殊，学习速率为0.0001的在测试集的精度达到了59%上下，学习速率为0.01达到了80%以上，学习速率为0.1 和0.001在75%左右。是因为它们收敛到不同局部最小值的缘故？业界普遍认为学习率为0.001较好，但是在这个网络中，0.01无论是在测试集正确率还是在收敛速度上均比0.001 强，可见学习速率的设置还与网络自身特性有关。

势能因子 为了加快网络的收敛速度，更快到达最小值，除了调整学习速率（步长）外，另一种策略是调整梯度。传统的最速梯度下降法在遇到曲率比较大的椭圆形优化平面时，梯度方向近乎和最优下降方向垂直，具有较差的收敛速度。引入势能法可以有效解决这个问题，势能法中梯度按照式25方式计算

$$g^{(k+1)} = m \cdot g^{(k)} - (1-m) \cdot \frac{\partial L_\theta(x)}{\partial \theta} \tag{25}$$

势能因子m的引入使搜索方向不至于过快变动，‘平滑’搜索的方向。由于存在势能，搜索时不至于会被无故拉入局部最优值。势能法在粒子群优化算法最为常见，势能的引入使粒子在移动过程中能够扩大搜索范围，不至于被卷入局部最优。

同样，我也对势能因子对网络收敛速度和网络性能做出一些小实验，调整势能因子，观察网络训练精度、测试精度和训练时损失变化，得到如图3所示结果。当势能因子等于1时，意味着

$$g^{(k+1)} = g^{(k)} = \cdots = g^{(1)} \tag{26}$$

搜索方向保持不变，如图3子图（1,1）（1,2）所示，注意到，按照单方向搜索时，损失函数波动起伏，具有非常多的最小值，这说明损失优化曲面存在很多‘坑洼’，必须采用某种平滑技术，使

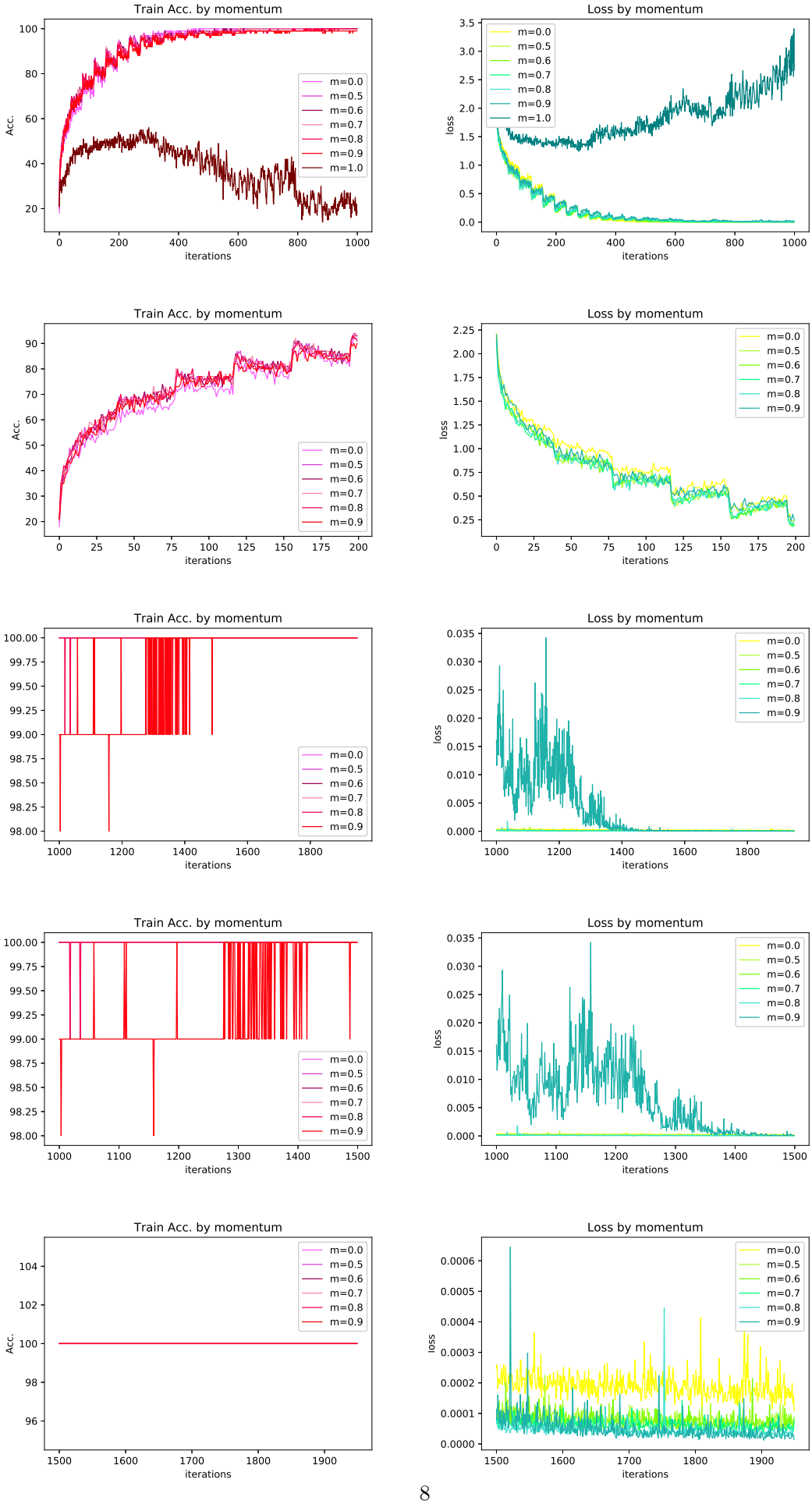

图 3: 不同势能因子下准确度和损失与迭代次数关系图

用mini-batch是一个平滑‘坑洼’的有效方法。

图3 中（2,1）（2,2）子图说明势能因子取0-0.9，在训练前期具有近乎相同的效果。在训练后期，当势能取0、0.5、0.6、0.7、0.8都基本收敛时，唯独势能取0.9的网络在波动，势能越大越有可能跳出局部最小值，扩大搜索的方向。有意思的是图3子图（5,2）出现了明显的分层现象，势能较大对于的曲线位于最下方，势能因子越小，收敛到的最优值越大。这也说明势能因子取0.9时，训练出来的网络性能最好。

深度 我在模仿原作者的实验时，观察到残差网络的加深并没有使网络的性能提升，反而使网络的性能下降。

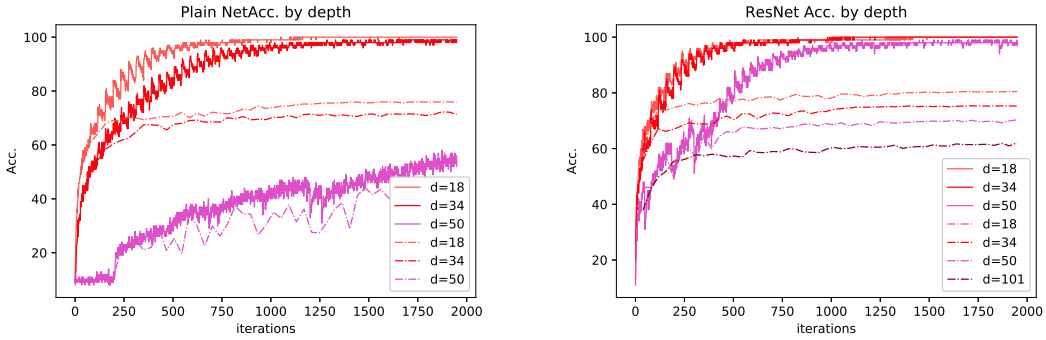

图 4: 不同深度下准确度与迭代次数关系图

图4左子图是普通网络精度和深度的关系，图4 右子图是残差网络精度和深度的关系，实线是训练精度，点线是测试精度。对比两图，我们发现两条规律，一是当深度加深是普通网络相比残差网络收敛速度更慢，更难训练，尤其是当深度为50时，其收敛速度明显慢于深度为50的残差网络。另外一条规律是深度越深的网络训练出来的性能越低，这是由于过拟合造成的，网络在训练集精度达到100%，而在测试集表现较差，越复杂的网络过拟合程度越高。

权重衰减 为了预防过拟合通常有三种手段，（1）扩大训练集样本个数（2）使用正则化（3）dropout。利用训练集已有的数据，通过镜像操作、平移操作产生一些人工数据以扩大训练样本数量，发现精度的确有所提高。这时候由于模型自身表现力不足，loss没有收敛到0。改进模型后，可以进一步提高正确率。使用L2正则化，在loss函数加上一个惩罚项来防止过拟合没有如同第一种方式那么明显，我尝试调整weight decay参数，得到如图5所示结果

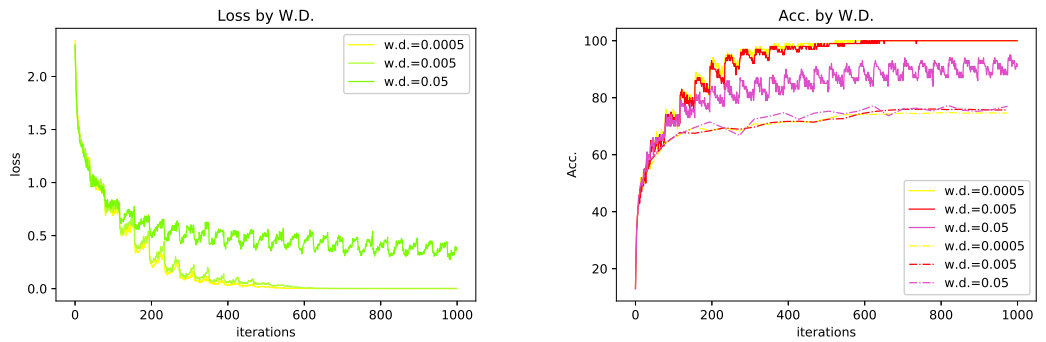

图 5: 不同权重衰减下网络性能与迭代次数关系图

weight decay取较小值时，网络具有基本相同性能，当weight decay取较大值时，损失函数难以优化到0，在训练集上的正确率下降，而且在训练后期无法收敛到固定值，而呈现周期性波动，不管weight decay取什么值，网络在测试集上的正确率相当。

过拟合 将训练样本图片镜像操作和平移变换后，训练样本更加具有代表性，测试使用resNet18训练后，测试集上正确率达到85%左右，提高了5个百分点，同时训练精度降到90%左右，这说明网络拟合力不足。我在搭建网络过程中，发现一个问题，原作者在论文中所介绍的网络适用于ImageNet数据集，输入为224×224，在CIFAR10上的图片均为32×32，如果不加修改直接套用，会导致在第一层卷积和池化层中，特征图的大小直接由32×32降到8×8，信息量丢失，我把第一层卷积和池化层的步长改为1，使特征图的大小只在第2、3、4残差层缩减。结果性能得到提高。实验结果如图6。

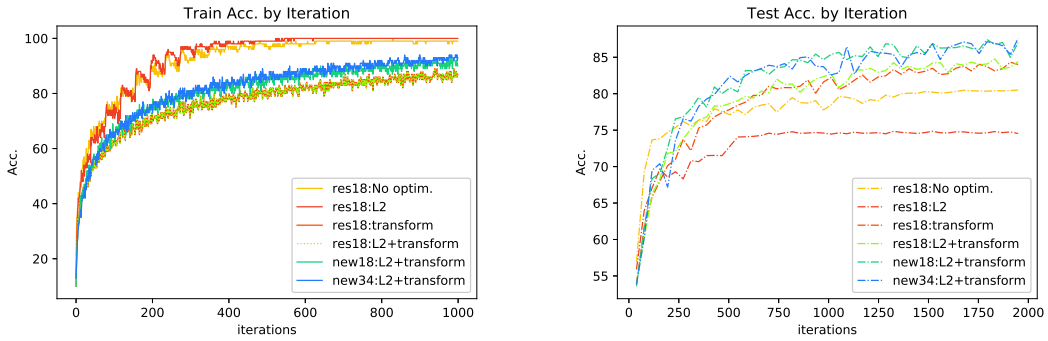

图 6: 不同权重衰减下网络性能与迭代次数关系图

图6中黄线使没有经过任何防止过拟合现象的18层残差网络，红线使用了L2正则化，权重衰减因子为0.0005，我们发现使用L2正则化并没有减轻过拟合现象。橙线使用了数据集增强，测试准确度提高，训练准确度降低，有效减轻了过拟合现象，绿线既使用了L2 正则化，也使用了数据增强，和只使用数据增强近乎没有什么差别。浅蓝线和深蓝线使用了改进的网络结构，并且同时使用L2正则化和数据增强，测试准确度提高，由于网络结构的拟合能力提高，所以相比使用防止过拟合策略的原网络，其在训练集上的精度提高。

参考文献

[1] Li Da, Li Lin, and Li Xiang. Classification of remote sensing images based on densely connected convolutional networks. In *IEEE Conference on Computer Vision & Pattern Recognition*, 2017.

[2] MartinT Hagan. *Neural network design*. 2002.

[3] D O Hebb. *The organization ofbehavior*. 1988.

[4] J. Nievergelt. R69-13 perceptrons: An introduction to computational geometry. *IEEE Transactions on Computers*, C-18(6):572–572, 2006.

[5] Walter Pitts. A logical calculus of the ideas immanent in nervous activity. *Bulletin of Mathematical Biology*, 52(1 – 2):99–115, 1990.

[6] F Rosenblatt. A probabilistic model for information storage and organization in the brain. *Psychological Review*, 386-408(65):6, 1958.