Project-2 of “Neural Network and Deep Learning”

陈亦雄 16307110231

一、前言

这篇报告主要涉及两个任务：首先是在 CIFAR-10 数据集上进行各种基于残差网络（ResNet）的网络结构的实验，试图找到一个最佳的模型结构以及训练方法，以及其中蕴含的规律；其次是使用 VGG 模型来探究批规范化（Batch Normalization，BN）对网络训练所起的作用。

这两个任务都会涉及各种网络结构、训练方法的改进和相关实验，但是前者更加偏向对训练策略、网络结构（卷积核尺寸、隐藏层大小等）的实验，并且期待于能够得到一些训练网络的实操经验。对于 VGG 的第二个实验而言，本文会专注于对批规范化方法的探索，对比是否包含批规范化层对训练结果的影响。两组实验都会有一系列的设定以及不同设定下对应的实验结果展示。最后，本文还包含对实验进行一系列的可视化，第一个任务具体涉及使用 TensorBoard 进行网络参数的分布可视化、第一层卷积核的可视化、展示模型错误分类的样本、训练过程中损失函数及准确率的变化曲线四者；第二个任务则会对是否包含 BN 层的神经网络进行训练过程的绘图对比。

二、在 CIFAR-10 上探究神经网络的训练方法

2.1 CIFAR-10 数据集及使用的神经网络介绍

CIFAR-10 是一个简洁的图像分类数据集，一共含有十类的 60000 张32×32低分辨率图片，其中每一个类别有 6000 个样本。这个数据集一般被分割为 50000 个训练样本和 10000 个测试样本，一个直观的可视化如图 1。

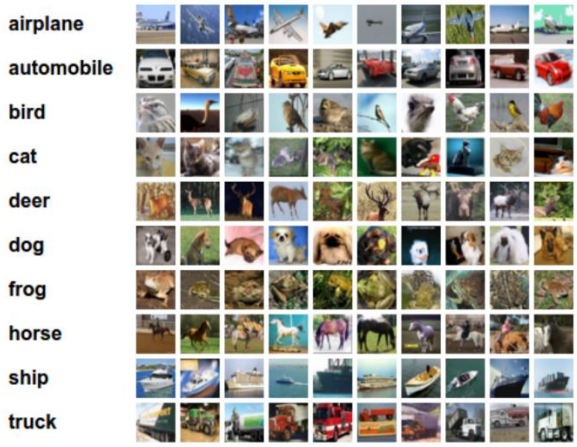

图 1. CIFAR-10 数据集十类样本可视化

由可视化我们可以看出，CIFAR-10 的分类目标很明确，各种图片之间也没有类别重叠或者类别包含的关系（汽车一类仅包含家用小型车，不包含最后一类卡车）。然而，由于图像的分辨率过低，一张图中也会存在语义比较模糊的现象，综合来看分类具有一定难度，

无法像类似的 MNIST 手写数字分类任务一样达到近乎 100%的准确率。实验初期，将分类性能的目标暂定为 90%准确率。

分类任务使用的模型选择的是 ResNet，这是一种包含残差结构的深层神经网络，利用残差模块来克服了深层网络难以训练的问题。具体而言，ResNet 解决的是网络参数的退化问题：随着网络深度增加，积矩阵的奇异值变得越来越集中，而小部分出现频率很低的奇异值变得任意的大。这种结果不仅仅和线性网络相关。在非线性网络中也会出现类似的现象：随着深度增加，给定层的隐藏单元的维度变得越来越低，即越来越退化。实际上，在有硬饱和边界的非线性网络中（例如 ReLU 网络），随着深度增加，退化过程会变得越来越快。通过引入残差模块（图 2），使模型每次训练的是当前网络块的输出与输入之差，强制性地让网络仅仅在原来网络层的基础上提升网络的表示能力，使得更加深层的网络训练结果几乎必定比更浅层的版本要好。

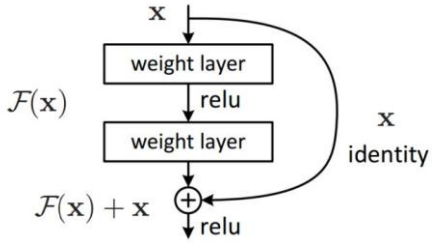

图 2. 残差模块

在 ResNet 的基础上，为了使得网络更加适应 CIFAR-10 的分类任务，这篇报告还尝试了基于这个模型的若干种改动，以及若干种训练策略。一开始的实验使用的是 PyTorch 框架 0.4.1 版本，ResNet34 模型训练 50 个 Epoch（一个 Epoch 意味着将数据集全部遍历一遍）耗时 2705 秒，后来改用 PyTorch 1.3.1 版本，50 个 Epoch 耗时 1105 秒（都为 1080Ti 单卡）。事实证明，相同的机器，使用不同版本的框架进行实验，开销会有比较大的差别，建议使用更新版本的框架，可能会有意想不到的收获。

实验的默认设置是将 ResNet 的第一层卷积改成了 3x3 且 stride=1 的形式（原来是 7x7 且 stride=2），这是由于对于小图片而言，7x7 卷积实在是太大了，提取出来的特征不够局部，更小的步长也可以尽量更加多地遍历图像中的位置。此外，默认学习策略为：学习率设置为 0.1，每经过 10 个 Epoch 下降为原来的 0.1 倍，SGD 优化器，动量参数 0.9，权重衰减（L2 正则化）系数 1e-5，批大小 128。以下的实验结果均为训练了 50 个 Epoch 后的结果，此时网络基本过拟合到 99%+的训练准确率。此外，实验并未设置随机数种子，所以相同设置下可能准确率结果可能有所差异（2%以内）。如果没有另外注明，则每一组实验的耗时都在 1100 秒左右。具体实验如下，每一组实验都是在默认设置的基础上进行更改的，以测试单变量对模型的影响。

2.2 实验

2.2.1 测试不同的第一层卷积核尺寸

第一层卷积核	最高验证准确率
3x3 + stride=1（对照组）	***78.62%
3x3 + stride=2	74.07%
5x5 + stride=2	72.53%

7x7 + stride=2（原论文中）	62.84%

经过实验，证实了原论文中第一层卷积核过大的事实。使用的卷积核尺寸从 7x7 逐渐减小为 3x3 的过程中，验证集的最高准确率单调上升，尤其是由 7x7 减小到 5x5 时上升尤为明显。这是由于卷积核主要的目的是提取图像的局部特征，而如果卷积核的尺寸相较图像而言过大，会导致提取出来的特征缺乏局部性。卷积神经网络的第一层（包括 ReLU）将原图像的局部 RGB 信息进行了第一次非线性变换，使得图像的边缘信息、纹理信息都得到了初步的转化，为后层进一步提取特征做了数据处理。而过大的卷积核显然是会严重忽略纹理信息的。试想，一个尺寸与原图相仿的卷积核在对图像做了卷积操作后，几乎只是将各个不同位置的一大块区域像素值直接求了加权平均罢了，并没有获得什么图像信息。

在将 3x3 卷积的步长设置为 1 后，最高验证准确率进一步显著上升了。当步长为 2 时，第一层卷积层的输出为 16x16 的特征图，而步长为 1 时特征图为 32x32，后者提取出来的信息更为细致、多样，对于这样的小图片而言增加提取的信息量并不会造成大的冗余，实验结果清晰地证明了这个设想。

为了更为直观地展示 ResNet 的训练结果，本文将原版 ResNet 的第一层 7x7 卷积核以及这一层卷积核的输出进行了可视化。如图 3、图 4 所示，我们可以发现 7x7 的卷积核呈现出一定的规律，每一个不同的卷积核事实上对应着一种图像中的模式，这个卷积核的目的就是将这个模式识别出来并编码成特征以供后层继续处理；同样的，我们发现第一层卷积层后输出的 32x32 特征图中，部分仍然可以看出原图（一只青蛙）的轮廓，然而另外一些则已经失去了人眼可以识别的特征。

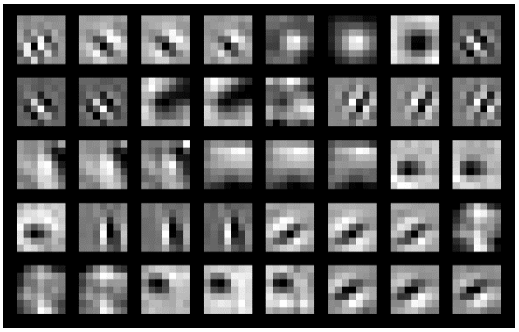

图 3. ResNet34 的 7x7 卷积核

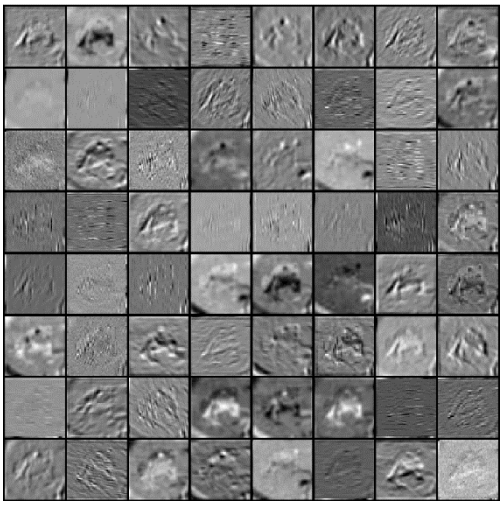

图 4. ResNet34 将青蛙图像进行处理后，第一层卷积层的输出

我们从第一组实验中选取最佳的网络设定：3x3 卷积核加上步长为 1，来进行接下来的实验，以避免实验准确率过低导致代表性不强。尽管 7x7 卷积可以有良好的可视化结果，但是由于其对小图像的编码能力太弱，故在后面的实验中不再讨论。

2.2.2 测试不同的数据增强方法在 CIFAR-10 数据集上的表现

数据增强方法	最高验证准确率
无（对照组）	78.62%
Color Jitter（四项参数都为 0.2）	74.95%
Random Crop（pad=4, size=32）	71.41%（0.4.1 版本 PyTorch）
Random Flip（0.5 概率，水平）	76.26%（0.4.1 版本 PyTorch）
Random Flip + Random Crop	69.12%（0.4.1 版本 PyTorch）
Random Erasing（0.5 概率，默认设置）	82.27%
0.2 概率随机将图片转换为灰度图	79.37%
随机灰度+Random Erasing	***82.53%

第二组实验主要探究了不同的数据增强方法对模型性能的影响。众所周知，深度学习模型本身对最终性能的影响固然巨大，但是影响更为广泛的是数据。数据的数量以及质量决定了不论使用什么模型能达到的性能上限。试想如果我们不让一个小孩接触足够高质量的教育、足够多的练习机会，又怎么能够期盼他成为社会的顶梁柱、某个领域的专家呢？方仲永就是一个很好的例子，即使天资再高（模型足够精妙），不接受足够的良好教育（接受的数据样本少，且质量低），必定是无法成才的。这是由于复杂的现实使得例外样本在总样本中占比不低，数据分布事实上是呈现长尾性质的。常见的几类样本在真实数据中占据大多数，而余下的成千上万种数据却不常见，如果没有良好的举一反三的能力（泛化能力）自然一旦遇到偏怪数据时就会显得无能为力。所以我们需要模型的决策边界尽量平滑且覆盖所有情况，以达到遇见不常见数据时不会过于固执（过拟合）的效果。

如表格所示，本文一共探索了五种常见的图像数据增强方法，以及两种增强方法的组合。由结果可见：1）如果将图像进行颜色抖动，改变原图的色相、饱和度、亮度等，原图会呈现出不同的颜色状态，此时分类性能反而下降。这可能是由于颜色抖动造成了训练数据中出现了现实中不可能出现的样本，例如紫色的青蛙、绿色的马等等，对模型进行了误导的结果。2）将图像进行随机裁剪会使得图像丢失部分重要信息，导致分类性能下降。3）随机水平翻转使得数据有了对称的样本，而不改变语义信息，尽管在当前设定下表现不佳，但仍是一种很好的数据增强方式。注意垂直翻转方法在当前任务下不可行，因为大多数情况下都会改变语义信息，例如没人见过卡车在天上开。4）随机遮挡在当前设定下是最好的单一数据增强方式，随机遮挡一块不大的区域使得网络在提取局部信息时对这一块空缺有了容忍能力，在以后遇到只包含类别物体的局部时就有更好的容错能力。5）随机将原图改变为灰度图直接使得这个 RGB 图像识别任务成为了跨模态任务，对模型的容量有比较高的要求，在当前设定下被证实为有一定效果。

不出所料，将图像随机裁剪和随机翻转进行结合后，两者原本都对模型起反作用，现在反作用更加明显了，模型精度进一步降低。而对于两种有效的数据增强方法，它们的互相结合进一步提升了模型性能。值得注意的是，以上得出的结论仅仅对当前的模型设定是成立的，不可以随意迁移到其他模型、其他数据集上。

2.2.3 测试不同的 L2 正则化

损失函数 + 正则化参数	最高验证准确率
CELoss + 0.00001（对照组）	78.62%
CELoss + 0.0001	80.16%
CELoss + 0.001	***82.69%
CELoss + 0.01	75.86%

L2 正则化是一种有效的正则化方法，能够明显地提升模型的泛化能力。这种方法通过在损失函数上添加参数的范数项，使得模型训练有缩小参数的倾向，最终得到一个参数绝对值较小的模型。在奥卡姆剃刀原则的假设下，其他条件相同时我们应该更加相信那个更简单的解释。于是我们也可以想象在训练准确率相同的情况下，我们应该更相信那个参数更小的模型。正则化对于模型参数缩小的影响可以通过调节正则化参数调节，这个参数越大，模型更加倾向于使得参数缩小。由于交叉熵损失函数是分类任务中几乎唯一的合适选择，于是我只通过改变正则化参数来实验不同的损失函数对模型的影响。

如表格所示，当正则化参数从 0.00001 逐渐增大到 0.01 时，模型的最高验证准确率是先上升后下降的。当正则化参数为 0.001 时，最高验证准确率达到了 82.69%，相比其他设定有显著提升。为了展示正则化参数对参数大小的影响，本文对无正则化以及考虑正则化（正则化参数为 0.01）的两个模型的第一层参数的分布使用 tensorboardX 进行了可视化，如图 5 所示。从对比图中我们可以很容易地看出相同模型有无 L2 正则化训练得出的参数分布是存在很大的差异的。有正则化时，模型参数分布十分紧密，维持在± 0.1的范围之内；而没有正则化时，模型参数分布较为分散，显然标准差大于有正则化的参数。

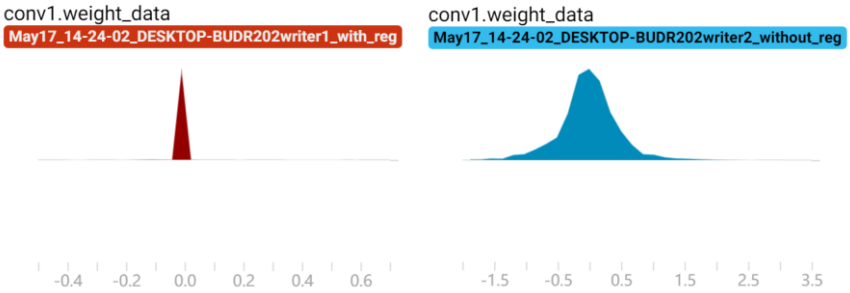

图 5. 有/无正则化的模型第一层卷积层参数分布对比

2.2.4 测试不同的 batch size

Batch size	最高验证准确率
16	80.89%（5000+秒）
32	***82.06%（2000+秒）
64	76.78%（1200 秒左右）
128（对照组）	78.62%（1100 秒左右）
256	79.05%（1000 秒左右）
512	75.99%（800 秒左右）

批大小（batch size）可以被视为最重要的超参数之一，它直接决定了一次参数更新迭代由多少不同的样本组成的样本空间决定。而它的这个特点又间接影响了训练收敛需要的迭

代次数以及最终的模型精度。经验之谈，一般批大小设置得越小，达到相同精度所需要的迭代次数越多，Epoch（一个 Epoch 为遍历一次所有样本）越少，最终的训练时间越长。而能够达到最佳精度的批大小普遍出现在 128 左右，故实验设计为由 128 为中央分界线，往两侧寻找最佳的参数。

因为 GPU 的特殊结构，批大小一般设置为 2 的指数，这样可以最大限度地使用 GPU 的并行计算能力。如上表所示，经过实验我们发现最佳的批大小为 32，此时最高验证准确率达到最大值，相比其余设定有明显提升。

2.2.5 测试不同的激活函数

激活函数方案	最高验证准确率
最后一层加 softmax	70.82%
所有层使用 ReLU（对照组）	80.60%
所有层使用 ELU	***82.26%
所有层使用 LeakyReLU	77.81%
所有层使用 RReLU	80.02%
所有层使用 Sigmoid	75.47%
所有层使用 tanh	74.66%

注意到 PyTorch 官方对 ResNet 模型的实现代码中最后一层全连接层后是不包含 softmax 激活函数的，于是将这一个激活函数加上后进行一次实验，我们发现这个模型相比没有 softmax 层的对照组有十分明显的退化。对于分类问题而言，softmax 函数能够使得网络最后的输出向量求和为 1，因此每一个输出值就可以当作是对应类的概率。然而在本实验设定下,加上这一层后网络反而性能下降。在查阅 PyTorch 文档后发现 nn.CrossEntropyLoss 类本身就是对数 softmax 层与负对数似然损失函数的结合，如果再在前面加上 softmax 函数，就相当于使用了两次 softmax。两层 softmax 函数会使得原神经网络的各输出值差距被剧烈拉大，使得最后计算出来的损失比只含有一层 softmax 层的情况更小，求得的梯度也更小，因此不利于梯度下降法更新网络参数。

在提取特征的网络层中，对照组使用的是 ReLU 激活函数。本文还尝试了 5 种不同的激活函数，它们分别是软饱和激活函数 Sigmoid, Tanh，以及硬饱和激活函数 ReLU 的 3 种改进——ELU, LeakyReLU,以及 RReLU。其中 ELU 相比于 ReLU 改进了原点附近的函数值，使之变得光滑；LeakyReLU 函数改进了 ReLU 的负半轴部分，使之有一个小的斜率，避免了死神经元的现象；RReLU 函数同样使得负半轴部分具有小的斜率，但是这个斜率是某个范围内的随机值。通过实验结果我们能够看出：ELU 激活函数表现最佳，最高验证准确率明显超过第二名。软饱和激活函数都表现不佳，最高验证准确率全部低于所有非饱和激活函数。令人吃惊的是，ReLU 在所有选项中排在第二，并没有因为其设计简单就弱于它的改进版本。

从这个实验中我们可以发现：1）对于深度神经网络而言，硬饱和激活函数的性能要好于软饱和激活函数。2）ReLU 的改进版本并未都在分类任务上超越原版 ReLU 函数，而应该批判地看待这一系列激活函数，各种改进版本可能各有各的用武之地。

2.2.6 测试不同的网络结构（卷积核数量、全连接层数量、dropout）

设定	最高验证准确率
卷积核数量 x2（此时输出 1024 维）	71.67%

原方案不变（对照组，此时输出 512 维）	79.64%
卷积核数量/2（此时输出 256 维）	***80.29%
卷积核数量/4（此时输出 128 维）	77.33%
输出 256 维+10 神经元隐藏层	***80.95%
输出 256 维+20 神经元隐藏层	80.44%
输出 256 维+30 神经元隐藏层	80.75%
输出 256 维+50 神经元隐藏层	80.58%
输出 256 维+100 神经元隐藏层	79.85%
输出 256 维+10 神经元隐藏层+dropout=0.1（dropout 放在 bn 前）	***81.31%
输出 256 维+10 神经元隐藏层+dropout=0.1（dropout 放在 bn 后）	79.64%（发现不如 dropout 在 bn 前）
输出 256 维+10 神经元隐藏层+dropout=0.2（dropout 放在 bn 前）	81.04%
输出 256 维+10 神经元隐藏层+dropout=0.3（dropout 放在 bn 前）	80.99%
输出 256 维+10 神经元隐藏层+dropout=0.4（dropout 放在 bn 前）	80.94%
输出 256 维+10 神经元隐藏层+dropout=0.5（dropout 放在 bn 前）	79.72%

这一组实验对原始 ResNet 网络进行了一系列改进的尝试，它可以看作三小组实验，每一组实验都在上一组的最优结果中继续改进。这三组实验的变量分别为 ResNet 网络中每一个特征图的通道数量、最后一层卷积层后添加的一层隐藏层的神经元数量、隐藏层的 dropout 概率。

如上表所示，首先改变了每一层卷积层的输出通道数，在所有设定中我们发现将卷积核折半能够得到最佳模型性能。实际上这一步的调整在很大程度上改变模型的参数量和计算量。如果忽略全连接层，则网络的运算量与每一层卷积核的数量成正比，卷积核的数量越少则推理速度与权重更新速度越快，代价是模型表示能力减弱。模型容量太小会导致欠拟合，而模型容量太大会导致过拟合，而究竟多强的模型表示能力对于 CIFAR-10 数据集而言最合适呢？实验结论是对原来的 ResNet34 的所有卷积层通道数减半。

其次，我们在最后一层卷积层输出的特征向量后加上一层全连接层，使得全连接层的数量由一层增加为两层。两层全连接层之间的隐藏层神经元数量的取值经过实验，发现在 10 个神经元时使得模型最佳。此时添加的神经元数量与最终网络的输出类别数量相同。

最后，本文为最后添加的隐藏层神经元赋予一定的概率失活，使其的正则化能力增强。经过实验，在失活概率为 10%时模型性能最佳。实际上，在 ResNet 模型已经添加了批规范化（BN）层的基础上，Dropout 层的必要性不高。经过前人的大量实验，证实了 Dropout 在与批规范化层共同出现时，几乎不会对网络有任何的增强效果，反而在一些应用场景会拉低网络的性能。

经过以上三组实验，我们取折半的通道数量、十个隐藏层神经元且有 10%概率失活为此组实验的最佳网络设置。

2.2.7 测试不同的网络结构（神经网络层数，其他设定使用第六组实验中最佳的）

层数	最高验证准确率
18	80.56%（371 秒）
34（对照组）	***80.93%（509 秒）
50	74.94%（749 秒）
101	73.08%（1300 秒）
152	68.69%（2000+秒）

本组实验为 2.2.6 的延续，同为调整网络结构，故采用上组实验中最佳的网络设定：取折半的通道数量、十个隐藏层神经元且有 10%概率失活。这组实验测试了不同网络深度对模型性能的影响。深度学习模型中，大众普遍认为更深的网络结构能够具有更好的信息编码能力。自从 ResNet 网络被提出伊始，得益于短路设计对于深度神经网络退化性的极大改善，深度网络开始真正走向现实。在常用的网络层数设置中，ResNet 共有 18、34、50、101、152 层五种设置，它们对应了基础残差模块、瓶颈残差模块的几种不同的组装方式。

如表格所示，34 层的 ResNet 达到了最佳性能，反而当网络加深时模型性能有下降的趋势。当网络达到了 152 层时，网络的分类准确率甚至下降了 12%之多。这是由于网络容量太大，而 CIFAR-10 数据集的图像太小且样本数量过低导致的。当小数据集由大型网络编码时，网络能够极快地适配训练样本的决策边界，对训练数据过拟合，不利于泛化性能的提升。我们注意到，其实网络在 18 层时就已经很好地掌握了当前的分类任务，在深度增加为 34 层时，其实并没有明显提升。他们之间的差异甚至可以用模型随机初始化导致的性能波动来解释。

综上所述，对于特定任务，我们应该寻找合适的网络结构，而并非一味追求更深更强的模型。一方面过拟合会导致性能下降、使用一系列正则化方法的难度增加，也会加长训练、推理时间，使得增加的开销毫无性价比可言。

2.2.8 测试不同的学习率调整方案（以原默认模型为基准）

学习率调整方案	最高验证准确率
常数学习率，LR=0.1	78.68%
常数学习率，LR=1e-2	77.52%
常数学习率，LR=1e-3	62.12%
常数学习率，LR=1e-4	52.12%
固定间隔调整学习率 (LR=0.1,stepsize=10,gamma=0.1,对照组）	***80.33%
固定间隔调整学习率 （LR=0.1, stepsize=5,gamma=0.33）	78.86%
指数衰减调整学习率 （LR=0.1，gamma=0.9）	76.68%
余弦周期调整学习率 (LR=0.1, T_max=10, eta_min = 0)	78.70%

本组实验探究学习率对模型性能的影响。学习率是影响模型训练时长、最终性能最重要的超参数，它决定了参数一步更新的大小。如果学习率较小，则参数更新慢，甚至无法更新；反之如果学习率较大，则参数更新快，但可能会无法学习到细节信息并在最终准确率附近不断以较大的程度上下波动，导致训练不稳定甚至无法收敛。

实验一开始尝试了常数学习率，发现当学习率从 0.1 开始下降到 0.0001 的过程中，最高验证准确率一直在下降。换言之，最大的 0.1 学习率使得训练最有效。这与一般的机器学习算法训练不一致，一般而言我们会在逻辑回归或者 XGBoost 算法中见到 0.01 或者 0.001 学习率表现最佳。这是由于使用图像作为本任务的输入数据，在进入神经网络之前会进行标准化。对于 CIFAR-10 数据集而言，其标准差为 0.25 左右，相比整数值的像素图小了 2-3 个数量级。如果网络参数的初始化策略不变，将神经网络的输入数据减小 2 个数量级后，此时每一次更新时求得的梯度都会相应地减小两个数量级，所以为了达到原图输入时的更新量，需要将学习率增大两个数量级。这就是为什么 0.1 的大学习率在本任务上是最好的常数学习率学习策略。

随后，实验尝试了一系列的学习率退火算法。固定学习率意味着参数更新步长不变，而当网络参数已经接近最优时，更小的学习率有助于网络学习到更加细节的知识，使性能增强。在固定间隔学习率下降策略、学习率指数衰减、学习率余弦周期调整三种方式中，最终固定间隔调整学习率使模型达到了最佳性能。三种算法具体的参数设定请参考上表。事实上，这个实验结果是没有调整学习率退火算法参数的结果，如果考虑上参数调整，三种学习率退火算法应当表现相近。因为当参数接近完美时，三种算法应当使得网络参数收敛到同一个最优点。

综上所述，对于深度学习而言，最佳的学习率选取策略应当是指定间隔调整学习率。在能够使得网络参数收敛到最佳点的前提下（排除常数学习率），我们应当选择超参数最容易调整的方法。而指定调整间隔的学习率下降策略能够通过预实验的方式，观察网络会在何时达到稳定期，在一段稳定期后进行学习率下降，反复几次，基本上就能够使得网络达到最优参数了。因此在大多数情况下，指定间隔调整学习率的策略应当被列为首选。

2.2.9 测试不同的优化器（LR=0.1，weight_decay=1e-5，其余参数维持默认）

优化器	最高验证准确率
SGD（对照组）	78.36%
Adagrad	76.71%
RMSProp（0.1 学习率失败，改为 0.01）	75.41%
AdaDelta	69.48%
Adam	***79.19%

本组实验探究不同优化器对模型训练的影响。优化器作为网络训练的引擎，起着至关重要的作用，尤其是对某些难以训练的模型以及复杂的任务而言。神经网络的参数更新借助着梯度下降法，而优化器决定的是梯度如何影响网络参数这一步的更新。SGD 是最简单的梯度下降算法，它仅仅由当前步梯度直接更新网络参数（尽管 PyTorch 中的 SGD 算法可以添加动量）。Adagrad 算法引入了二阶动量，使得经常被更新的参数能够更新得慢一些，而偶尔更新的参数能够以更大的学习率学习到更多知识。RMSProp / AdaDelta 算法在 Adagrad 的基础上只考虑过去某个时间窗口内的二阶梯度，不累积全部历史梯度，克服了学习提前结束的问题。Adam 结合了一阶动量和二阶动量，使得模型参数更新受到了适应性的一阶加速度和二阶加速度影响。

在这五种优化器的最终实验结果中，我们发现 SGD, Adam 两种方法表现最佳。结合许多网络资源和先人经验，SGD 以及 Adam 被公认为是两种最优的优化算法，它们对各种任务有很强的鲁棒性，且不需要过度考虑参数调节问题，直接使用自定义的初始学习率并使其余参数保持默认即可达到很好的训练效果。对于相同的任务而言，相同的模型应当有一个最佳收

敛点，而 SGD（考虑动量）以及 Adam 算法能够最稳定地收敛到该点。相较而言，Adam 的收敛速度在某些情况下略快于前者，但一般两者的表现不会有很大差异。

2.2.10 对以上实验的小结

经过了以上的丰富实验，我们得出了以下分别达到最优的参数：

网络结构	学习策略
第一层卷积选用 3x3 + stride=1	随机灰度数据增强+随机遮挡数据增强
ELU 激活函数	0.001 的 L2 正则化系数
10 神经元隐藏层	批大小为 32
卷积层输出 256 维（每个卷积层为标准 ResNet 一半输出通道数）	固定间隔调整学习率（初始 0.1 学习率，每 10 个 Epoch 下调至原来的十分之一）
10%概率 dropout（dropout 放在 bn 前）	Adam 优化器

现在尝试对原来的默认网络设定采用上面得出的最佳网络结构以及最佳学习策略，如下表所示。我们对比默认参数的网络取得的 78%-81%之间波动的准确率，不论是使用最佳网络结构还是最佳学习策略都对网络有着不小的提升。但是如果将这两项进行组合，我们发现网络的提升不如只采用最佳学习策略的结果。事实上也是如此，单项由默认设置而来的最佳改动叠加在一起，并不一定是全局最优选择。在最佳网络结构的基础上，与它适配的学习策略相较于默认模型就会有变化。

所以我们从这个实验可以得出这么一个道理：如果希望追求全局最优设定，我们唯一的方法就是进行大规模网格搜索。如果仅追求局部最优设定，我们可以使用先前找到的最优参数来设定模型，在此基础上尝试下一组参数。而寄希望于将每一个参数分别的最佳取值组合起来，从而得到总体最佳模型，毫无疑问是刻舟求剑的行为。

实验设定	最高验证准确率
1 同时使用最佳网络结构以及最佳学习策略	82.64%（1997 秒）
2 只使用最佳网络结构	82.08%（516 秒）
3 只采用最佳学习策略	83.16%（1931 秒）

2.2.11 进一步改进 ResNet

经过查找网络资料后，对比而言自己设计的默认模型准确率太低，即使是调整过参数的网络也无法达到一开始预期的 90%准确率。经过启发，在 ResNet 的第一层卷积后，不该使用池化层。如果有池化层的话，在第一层卷积后，特征图就会由32×32减小为16×16,在比较浅层的时候就会损失大量信息，去掉这一层池化以后再进行试验，结果如下：

实验设置	最高验证准确率
1、 ResNet18, BatchSize=128, LR=0.1 SGD, 随机裁剪翻转数据增强, 5e-4 正则化系数 第 135,185 Epoch 学习率下降为原来 10%	95.45%（5938 秒）
2、第 1 组加上 10 神经元隐藏层、10%dropout	95.12%（5945 秒）
3、第 1 组使用随机遮挡、随机灰度数据增强	89.11%（5937 秒）

4、第 1 组使用 ResNet34 而不是 ResNet18	***95.48%（10286 秒）
5、第 1 组使用 ELU 而不是 ReLU	92.26%（5979 秒）
6、第 1 组使用 0.001 正则化系数	94.63%（5919 秒）

从表中我们很容易看出，第一组实验相较于之前的默认设置仅仅改动了第一个池化层，甚至还使用了容量更小的 ResNet18 来节省实验时间（将池化层去掉以后网络的参数增加，故训练实验也变长了）。此外唯一不同的设定就只有使用了更长的时间进行训练，共训练 240 个 Epoch。然而在此设置下，模型的最高验证准确率直接提高了 15%之多。这说明对于神经网络而言，尽量改动结构使其匹配任务能够取得重大的提升效果，其收效是远高于寻找最优参数的。

此外，本文还使用了先前实验中的部分最佳配置来修改这组实验中的默认配置，分别是加一层隐藏层、使用 dropout、使用随机遮挡、随机灰度数据增强、使用 34 层的 ResNet、使用 ELU 激活函数而不是 ReLU、使用 0.001 的正则化系数（第 2 到 6 小组实验）。最终发现仅仅有增加网络层数至 34 层有极其微小的性能提升，其他改动都会或大或小地影响网络性能。这更加印证了 2.1.10 中得出的结论：某设定下对一个参数的实验只能在这个设定下生效，无法代表普适情况。

本文记录下了表中第 6 小组实验最后的训练、验证准确率和损失函数随着时间变化的曲线，如图 6 所示。我们可以清晰地看出，当学习率一开始为 0.1 时，验证准确率大约能够达到 80%出头；在第 135 Epoch 处，学习率第一次降低，验证准确率迅速上升至 90%以上；在第 185 Epoch 处，学习率第二次降低，验证准确率达到了 95%左右，而训练准确率已经为 100%，所以没有继续减小学习率的必要了。损失函数同样遵循着相同的规律，只是不如准确率曲线那么直观罢了。值得注意的地方是，在第 150 Epoch 后，我们能够从上升的损失以及下降的验证准确率看出模型此时过拟合开始加重。

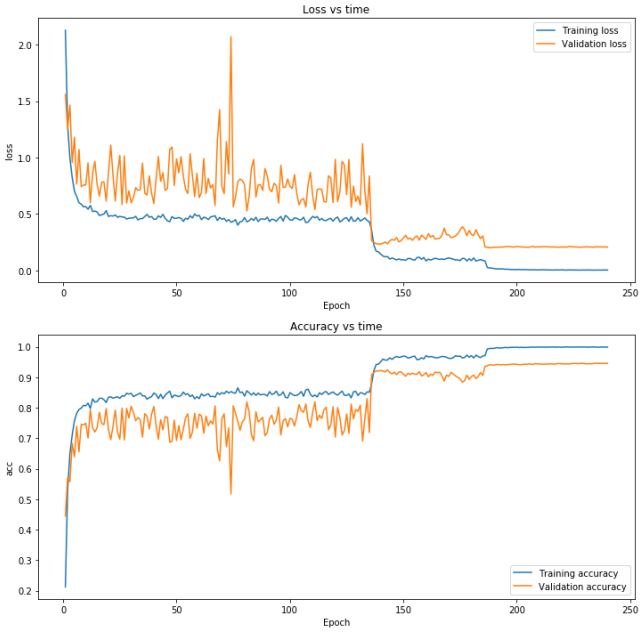

图 6. 训练、验证的准确率与损失函数变化曲线

2.2.12 案例研究

本文将最终训练得到的最优模型（2.2.11 中改进后的 ResNet18，最高验证准确率达到

95.48%）在测试集上的分类错误样本进行了可视化，如图 7 所示。从图中我们可以看出，模型的分类能力已经达到了一个很可观的程度，仅有的不多的错误分类结果都是一些模棱两可的图像。这些图像被误分类的类别一般不是与真实标签差异很大的类别。在图中我们可以看到，第一张古董汽车的图片的确容易因为它的平头而被认为是卡车；第二张 B2 隐形轰炸机的拍摄角度很难让没有见过这个型号的人认为它是飞机；而第三张图片甚至连人都无法分辨到底是什么；几张错分的猫狗由于图像清晰度过低，无法分辨面部特征，显得模棱两可。最优模型仅在一些难样本上犯错，达到了较好的分类结果。

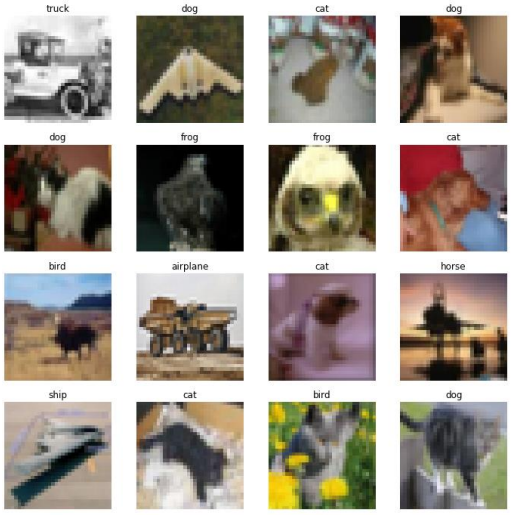

图 7.改进后的 ResNet18 的错分样本

由此可见，在 CIFAR-10 数据集上，现有的模型已经可以很好地完成分类任务。其分类能力与人类相比已经差异不大。这也是为什么在 2015 年以后提出的分类模型几乎已经不再考虑 CIFAR-10 数据集作为挑战目标，而专注于 ImageNet 这种大且难的数据集。尽管 ImageNet 已经包含很多样本，然而对于 1000 分类任务而言仍然算得上是挑战性十足，人类在其上的表现都至少会有 5%左右的 top5 分类错误率。而自从 SENet 在 2017 年的最后一届 ILSVRC 竞赛上拔得头筹，将 ImageNet 的 top5 错误率降至 3.79%开始，普通的图像分类任务已经不再是当前计算机视觉科研的主要任务。

三、使用 VGG 模型探究批规范化(Batch Normalization, BN)

3.1 BN 层简介

在一般的卷积神经网络中，批规范化层一般使用在卷积层后、激活函数前或者全连接层后、激活函数前。这一层的功能是对前一层的输出进行均值为 0，标准差为 1 的规范化，使得数据输入下一层时不会导致梯度消失或梯度爆炸，避免了网络无法训练的问题。具体而言，卷积层的批规范化的公式如下

$$O_{b,c,x,y} \leftarrow \gamma_c \frac{I_{b,c,x,y} - \mu_c}{\sqrt{\sigma_c^2 + \epsilon}} + \beta_c \qquad \forall b, c, x, y.$$

其中μ_c, σ_c为每一个通道上所有输入数据的均值以及标准差，γ_c, β_c为两个可学习的参

数，用于对输出数据做仿射变换。对于全连接层而言，操作也十分类似，只是变为对同神经元输入的所有样本数据做规范化。在训练时，对一批数据求出均值和标准差，而测试时对训练时输入的所有数据求均值和标准差。以下的实验着力于探索 BN 层在神经网络中具体起了哪些作用。

3.2 有无 BN 层的 VGG-A 模型对比

如图 8 所示，有 BN 层的 VGG-A 模型训练收敛更快，且最终达到的验证准确率高于没有 BN 层的 VGG-A 模型。在同为使用 0.001 学习率、Adam 优化器、无 L2 正则化、批大小为 128 的设定下，20 个 Epoch 后两个模型的训练准确率都接近 100%，然而有 BN 层的 VGG-A 模型的最高验证准确率为 82.84%，比没有 BN 层模型的 77.07%高了近 6%。这个现象可以说明在添加了批规范化的处理后，网络似乎有了正则化解决过拟合的效果。

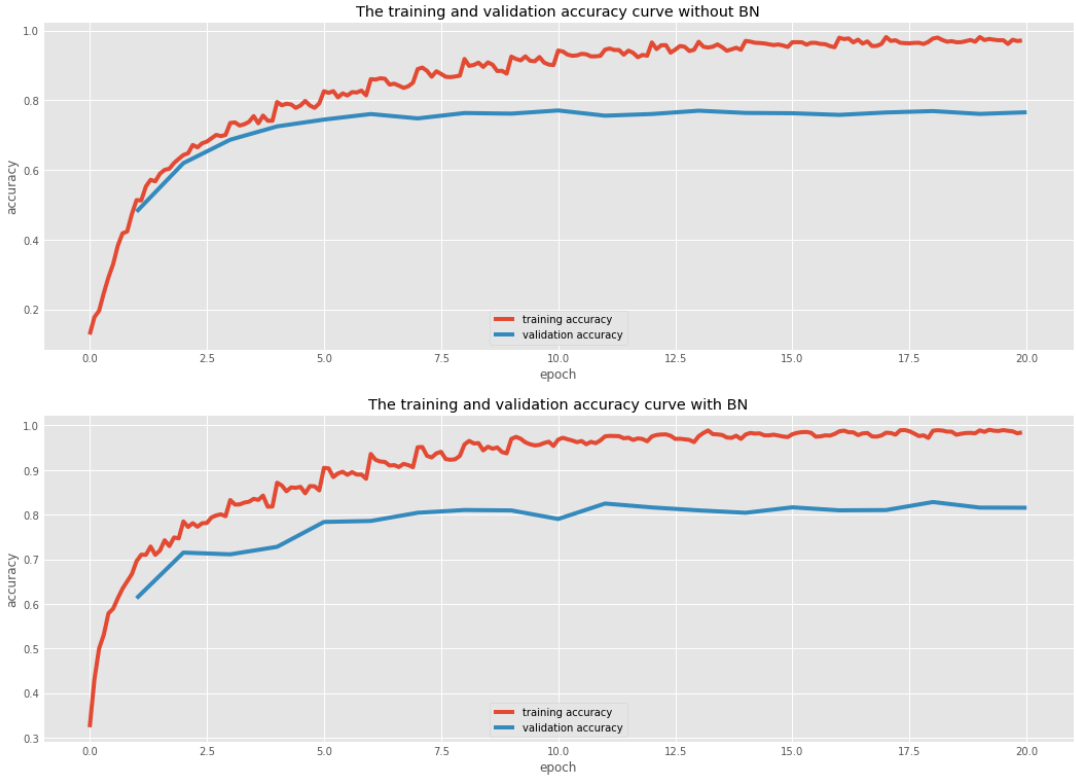

图 8. 是否有 BN 层的 VGG-A 模型准确率对比

以下几个模块中所做的实验除学习率以外，其余所有训练参数设定都与本组一致。

3.3 探究 BN 层是如何提升优化效果的

3.3.1 损失函数值

本文使用 VGG-A 模型（拥有 11 层权重层）以及 VGG-19 模型（拥有 19 层权重层）来探究是否使用 BN 层对训练损失函数下降的影响。一般而言，容量越大的模型在训练相同的迭代次数时能够使得损失函数下降得越多。由 3.2 的实验结果我们得知，不论是使用批规范化的模型还是不使用批规范化的模型，在训练达到了 15 Epoch 左右时都可以达到 99%左右

的训练准确率。尽管这是过拟合的结果，但是训练时能够快速过拟合同样也能够说明这个模型能够对这个任务进行快速适配。在本节中，我们使用有批规范化的模型以及没有批规范化的模型进行记录训练损失函数随着训练时间的下降状况，使用 0.0001、0.0002、0.0005、0.001、0.002 五种不同的学习率来分别训练 5 个模型，并记录它们中在相同训练步数时达到的最大损失以及最小损失，最后将其在同一张图中进行可视化，如图 9 所示。

从图中，我们可以很明显地看出，在前半训练过程中不论采用何种学习率，有 BN 层的模型都具有更小的损失函数值。这表明在具有 BN 层时模型的收敛明显要比没有 BN 层时更快。在后半训练过程中，两组模型可以近乎收敛到相同的训练损失函数值，即接近 0，表明此时两组模型都对训练数据集有足够强的表示能力，故才能对训练集近乎完美拟合。此外，我们还注意到在包含了 BN 层后，不同学习率的模型损失函数值差异度小，这体现在图中红色区域的宽度显然要小于绿色区域的宽度。我们可以推断，在包含 BN 层后，损失函数的波动性减小，训练的稳定程度上升了。

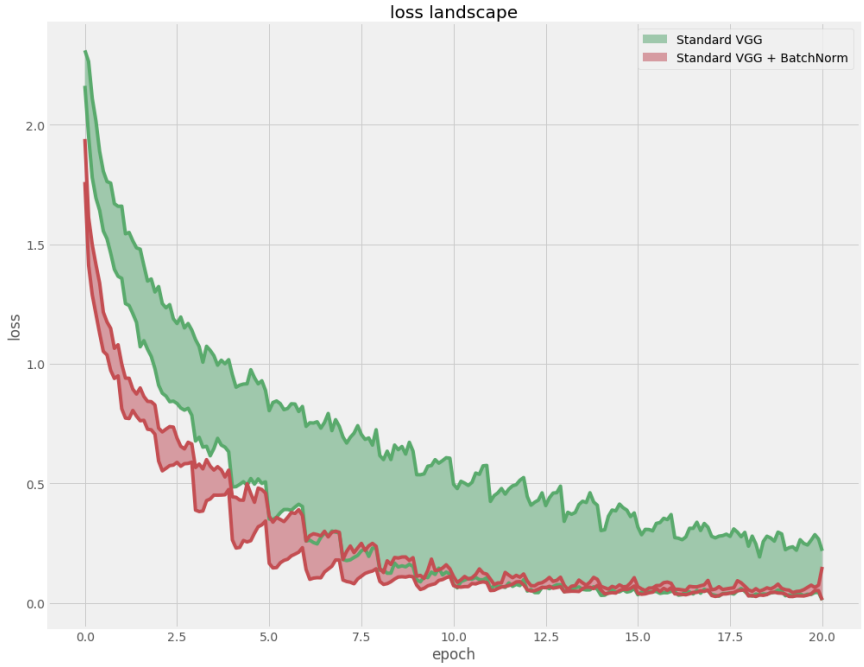

图 9. 是否包含 BN 层的模型训练损失对比

另外，本文还实现了 VGG-19 模型用于再次验证以上分析得出的结论。然而出乎意料的是，在经过使用上述实验相同设置重复的 VGG-19 以及包含 BN 层的 VGG-19 模型却并不能复现以上结论。当学习率为 0.0001 时，两个模型都可以正常训练，包含/不包含 BN 层的模型分别得到了 75.66%、73.27%最高验证准确率，不包含 BN 层反而最终模型性能更佳。当学习率为 0.0002 或更高时，不包含 BN 层的模型训练失败，验证准确率始终保持为 10%左右。初步推断这是由于梯度消失/梯度爆炸导致的，具体最高验证准确率的实验结果如下表。

学习率	VGG-19（不含 BN 层）	VGG-19（包含 BN 层）
0.0001	75.66%	73.27%
0.0002	10.00%	79.51%
0.0005	10.00%	84.12%

0.001	10.00%	82.47%
0.002	10.00%	85.00%

VGG-19 模型有 19 层权重层，已经能够算是深度神经网络了。在深度神经网络中，梯度消失/梯度爆炸是十分常见的现象。尽管模型已经由 Xavier 初始化进行了良好的参数初始化，但是随着训练的推进，参数的大小必定会发生改变，从而使得梯度随着网络的传播渐渐偏向使得网络难以训练的量级，尤其是网络前几层更是梯度消失的重灾区。观察训练过程，训练准确率在 10%上下不断波动，而验证准确率始终精确地保持在 10%，这个现象初步推断是梯度消失的特征。训练已经停止，训练准确率在不同时刻波动的原因是数据读取时每一批是随机生成的，故会有不同的准确率。观察每一个 Epoch 的损失函数，果不其然已经停止了变化。从包含 BN 层的模型能够良好训练，我们可以得出结论：BN 层使得模型的每一层数据流限制在一个合理范围内，保证了深层模型训练的稳定性。

最后，本文对加上 dropout 的 VGG-A 模型以及包含 BN 的 VGG-A 模型进行了实验，试图比较这两种模型正则化层哪一种有更好的效果。Dropout 加在模型最后两层全连接层上，其余部分保持不变；BN 层则添加在每一层权重层后。实验结果如图 10。

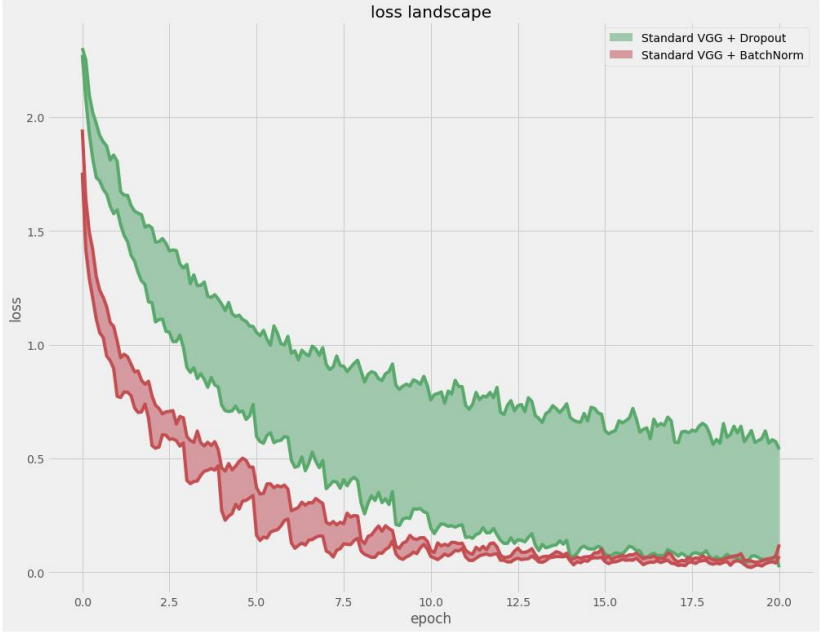

图 10. 包含 Dropout 层与包含 BN 层的模型训练损失对比

我们从图 10 中可以看出，包含 Dropout 的 VGG-A 的训练损失甚至比不包含此结构的 VGG-A 波动性、绝对值还要更大。这是由于考虑 Dropout 时，每一次训练都会导致神经元随机失活，不同的迭代步失活的神经元不相同，这使得最后求得的输出值会存在很大的波动。这个波动使得在训练时包含 Dropout 的模型看起来不那么稳定，损失函数也不会下降得很低。但是当最后测试时将所有可能的模型进行 ensemble，测试损失函数值相比于添加了 BN 层的模型，差距并没有想象中如此巨大。具体的最高验证准确率如下表：

学习率	VGG-A（包含 Dropout 层）	VGG-A（包含 BN 层）
0.0001	75.10%	73.55%
0.0002	76.93%	77.35%

0.0005	77.14%	81.82%
0.001	75.08%	82.96%
0.002	69.83%	81.55%

可以从表中看出，包含 Dropout 层的 VGG-A 模型在小学习率时性能比包含 BN 层的模型强，而当学习率增大以后，其性能反而开始严重衰退。综合所有情况来看，包含 Dropout 的模型最高性能与不包含 Dropout 的模型相比差异不大，并没有明显的提升，而其训练的稳定程度差于原始 VGG-A,与包含 BN 层的模型更是差距明显。由此我们可以得出一个结论：对于模型正则化层而言，BN 的性能在分类任务上要好于 Dropout 机制。

3.3.2 预测梯度

这一小节，我们的重心放在探索是否包含 BN 层的模型的梯度预测性上。具体而言，我们使用模型最后一层（输出层）的梯度，计算当前步与前一步的梯度差异（使用两者差的 2 范数来衡量）。如果两步之间的梯度差异较小，则说明模型的特征空间较为平滑，下降方向稳定，不会存在反复变动的情况。此时模型更加易于训练，更加容易得到较高的最高验证准确率。我们仍然使用 0.0001、0.0002、0.0005、0.001、0.002 五种不同的学习率来训练 5 种不同的 VGG-A 模型，取各个模型在当前步上的最大梯度差异与最小梯度差异并用同一张图进行展示，对比是否包含 BN 层的模型在梯度预测性上的表现。

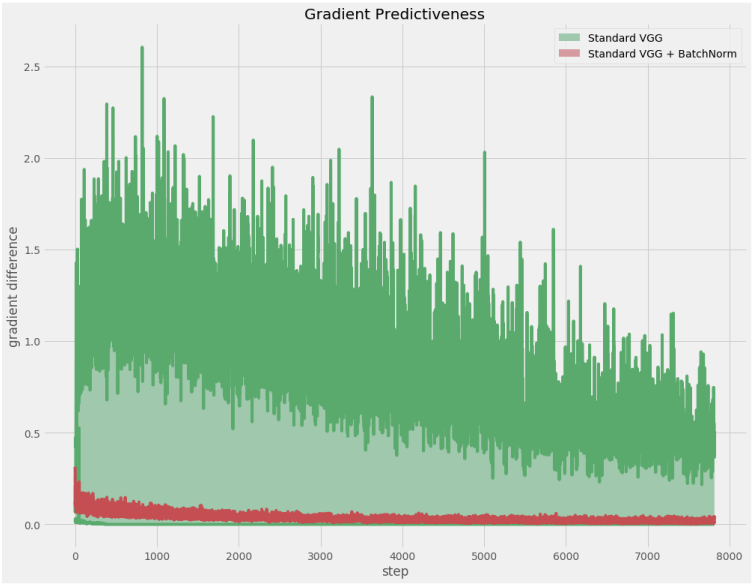

图 11. 是否包含 BN 层的模型的梯度预测性对比

如图 11 所示，我们可以看出当模型包含了 BN 层以后，相邻两步之间的梯度差异远小于不包含 BN 层的模型。这可以说明当模型包含了 BN 后，参数的优化空间更加平滑了，这也可以解释为什么在 3.3.1 中包含 BN 的模型损失下降得更快且更加稳定。此外我们能够看出，随着训练进度的推移，两组模型的梯度差都会逐渐降低。这是由于随着训练逐渐收敛，模型的参数会越来越收敛到优化空间的平坦之处，每一次反向传播的梯度本身会越来越小。梯度逐渐减小，使得更新越来越小，这才意味着模型逐渐接近收敛。这就是为什么两步之间的梯度差也会呈现减小的趋势。

3.3.3 有效β平滑度

有效β平滑度（effective β smoothness）指的是模型当前某一步的更新方向上更新不同的步长所得到的下一步梯度的最大差异（用差的 2 范数来衡量）。它可以用来衡量损失下降方向上函数优化空间的一阶、二阶平滑程度。由于原文中的定义在实现上较为复杂，于是本文使用不同学习率的模型在同一步上的最大梯度差异来代替。与前文实验类似，使用 0.0001、0.0002、0.0005、0.001、0.002 五种不同的学习率来同时训练 VGG-A 模型，并计算每一步中这 5 个模型的梯度最大差异，画出折线图。比较是否含有 BN 层的模型得出的结果，如图 12 所示。

我们从图 12 中可以看出，包含 BN 层的 VGG-A 具有明显更小的梯度差异，这说明它的有效β平滑度远远大于无 BN 层的 VGG-A。这与前文的实验结论互相印证，它表明 BN 层能够很大程度上平滑函数的优化空间，使得模型的优化更为顺畅。同时，我们也看出随着训练进行，无论是包含 BN 层还是不包含 BN 层的 VGG-A 模型都拥有越来越小的梯度差异，同时拥有越来越大的有效β平滑度。这可以说明当模型逐渐趋近于收敛，参数更新越小，且参数更新所能到达的优化空间越平滑。

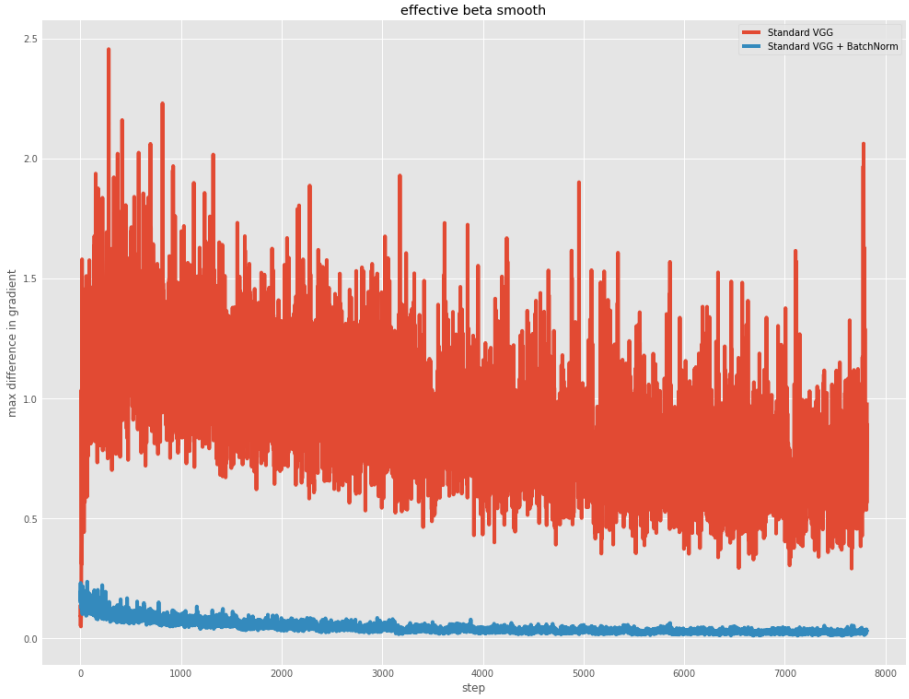

图 12. 对比是否含有 BN 层的 VGG-A 模型的有效β平滑度

四、结论

本文一共探究了两个问题：神经网络的各项调整在图像分类任务上分别表现如何、批规范化层在神经网络中起了什么作用。

首先对于第一个问题而言，本文得出了以下结论：1）对于图像分类问题而言，如果图像的分辨率较低，在首层使用3×3卷积是一个很好的选择。2）一般而言随机遮挡、随机翻转、随机裁剪是很通用且有效的数据增强方式，此外根据具体任务也可以尝试其他增强方式。3）L2 正则化系数、批大小是两个需要尝试才能得出最优值的数值型超参数。4）ReLU 基本上是最有效的激活函数之一，如果有特殊需求才需要对其他激活函数进行尝试。

5）改变网络的结构对网络提升巨大，可以在某些场景下大量节省训练及推理时间或者大幅提升网络的性能，需要具体情况具体分析。6）网络的深度并不是越深越好，在达到了当前任务的网络容量的前提下越浅越好。7）学习率调整策略建议优先使用手工设置间隔的学习率下降方法。8）优化器使用 SGD（含动量）或者 Adam 即可。9）如果希望找到模型的最优超参数，需要在找到一部分找参数的前提下用该最佳网络再去尝试下一个的超参数，而不该找到所有超参数的最佳设置再将其同时使用。

在探究第一个问题的过程中，本文也在 CIFAR-10 数据集上实现了一个性能较好的 ResNet 模型。它的主要改动有两处——将第一层卷积层从3×3卷积改为了7×7卷积、去掉了第一层卷积层后的池化层以增加模型容量。最后此模型的 34 层版本达到了 95.48%的最高验证准确率。其在一块 1080Ti GPU 上训练 240 个 Epoch，共耗时 2.86 小时。

其次对于第二个问题而言，本文得出了以下结论：1）包含 BN 层的 VGG-A 模型在 0.001 的学习率下训练最高得到了超过不含 BN 层的模型近 6%的测试性能，在训练准确率都接近 100%的前提下表明 BN 层有预防过拟合的正则化效果。2）BN 层使得模型训练更稳定，这体现在不同学习率的模型拥有相比无 BN 层更小范围变动的、下降更快损失函数曲线。3）加上 BN 层后，模型的梯度预测性与有效β平滑度都显著上升，这两个指标表明了模型函数的优化空间更加平滑，解释了为何模型的训练能够更加稳定。