

Thinking Deeper about Cifar10

方桂安, 刘梦莎, 刘玥, 罗秋琳, 马梓场, 唐迅

摘要—本次作业目标是对著名数据集 cifar10 进行分类, 我们使用 ResNet 作为模型, 采取了各种数据增广, 正则化等技巧来优化结果。并对类别不均衡、部分标注等问题进行研究, 通过阅读文献解决问题, 最后成功复现了 cifar10 的 sota 结果和前沿 autoML 的实践。

关键词—cifar10, 深度学习, 数据增广, 正则化

I. 概述

CIFAR10 数据集 (Canadian Institute for Advanced Research, 10 classes) 是 Tiny Images 数据集的子集, 由 60000 张 32×32 彩色图像组成。这些图像的标签来自下面 10 个互斥类别: airplane, automobile, bird, cat, deer, dog, frog, horse, ship, truck。每类有 6000 张图像, 其中包括 5000 张训练图像和 1000 张测试图像。

我们的报告将从模型 (ResNet)、优化技巧 (数据增广、正则化、梯度裁剪、Dropout、学习率衰减等)、类别不均衡问题、部分标注问题、延伸与拓展五个方面展开, 并在最后的结论部分对结果进行总结与思考。

II. RESNET 残差网络

A. 残差网络定义

残差网络能让非线性层满足 $H(x, w_h)$, 然后从输入直接引入一个短连接到非线性层的输出上, 使得整个映射变为

$$y = H(x, w_h) + x$$

这就是残差网络的核心公式, 换句话说, 残差是网络搭建的一种操作, 任何使用了这种操作的网络都可以称之为残差网络。

方桂安, 20354027, (e-mail: fanggan@mail2.sysu.edu.cn)。

刘梦莎, 20354091, (e-mail: liumsh6@mail2.sysu.edu.cn)。

刘玥, 20354229, (e-mail: liuy2236@mail2.sysu.edu.cn)。

罗秋琳, 20354095, (e-mail: luqilin3@mail2.sysu.edu.cn)。

马梓场, 20354103, (e-mail: mazy23@mail2.sysu.edu.cn)。

唐迅, 20354121, (e-mail: tangx66@mail2.sysu.edu.cn)。

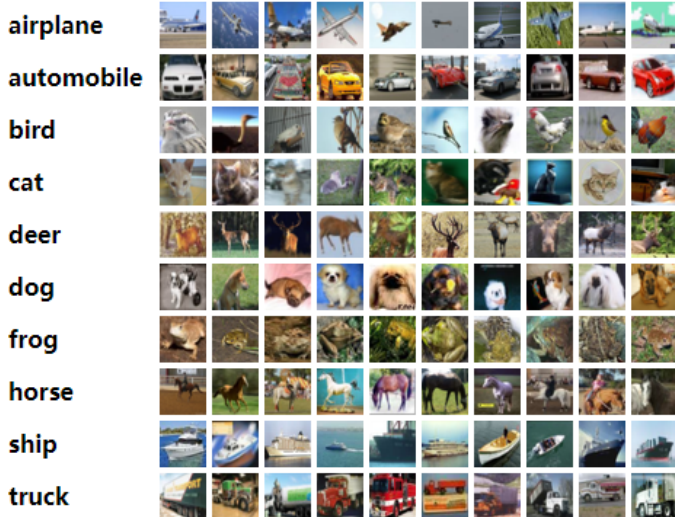


Fig. 1. cifar10 数据集

B. 残差结构

如图 2所示, 我们在残差结构中使用一个非线性变化函数来描述一个网络的输入输出, 即深层的输入为 x (x 也为浅层的输出), 深层的输出为 $F(x) + x$, F 通常包括了卷积, 激活等操作。

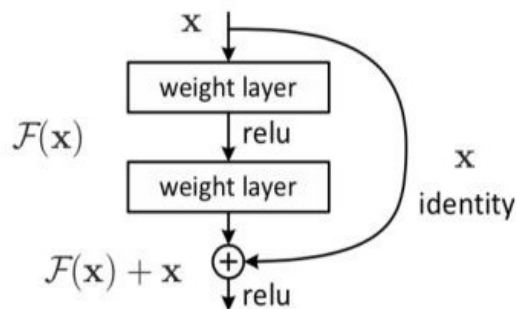


Fig. 2. 残差模块

这里需要注意附加的恒等映射关系具有两种不同的使用情况: 残差结构的输入数据若和输出结果的维度一致, 则直接相加; 若维度不一致, 必须对 x 进行升

维操作，让它们的维度相同时才能计算。升维的方法有两种：

- 直接通过 zero padding 来增加维度 (channel)
- 用 1x1 卷积实现，直接改变 1x1 卷积的 filters 数目，这种会增加参数

令 $H(x) = F(x) + x$ ，即 $H(x)$ 为深层的输出，则 $F(x) = H(x) - x$ 。此时残差结构如图 3 所示，虚线框中的部分就是 $F(x)$ ，即 $H(x) - x$ 。

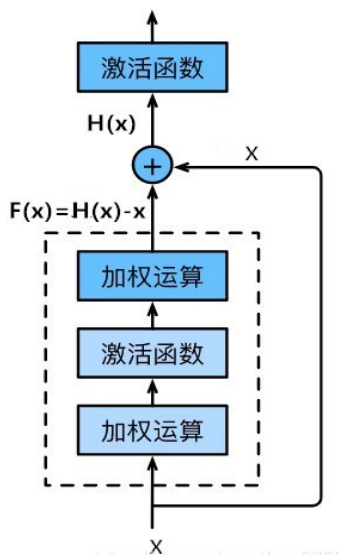


Fig. 3. 残差结构

当浅层的 x 代表的特征已经足够成熟（即浅层网络输出的特征 x 已经达到了最优），如果任何对于特征 x 的改变都会让 loss 变大的话， $F(x)$ 会自动趋向于学习成为 0， x 则从恒等映射的路径继续传递。这样就在不增加计算成本的情况下达到了以下两个目的：第一，在前向过程中，当浅层的输出已经足够成熟，让深层网络后面的层能够实现恒等映射的作用（即让后面的层从恒等映射的路径继续传递）；第二，解决了由于网络过深而导致的模型退化的问题。

另外，残差结构可以让网络反向传播时信号可以更好地传递，以一个例子来解释。

假设非残差网络输出为 $G(x)$ ，残差网络输出为 $H(x)$ ，其中 $H = F(x) + x$ ，输入的样本特征 $x = 1$ 。

- 1) 在某一时刻，非残差网络 $G(1) = 1.1$ ，把 G 简化为线性运算 $G(x) = W_g * x$ ，可以明显看出 $W_g = 1.1$ 。残差网络 $H(1) = 1.1$ ， $H(1) = F(1) + 1$ ， $F(1) =$

0.1，把 F 简化为线性运算 $F(x) = W_f * x$ ， $W_f = 0.1$

- 2) 经过一次反向传播并更新 G 和 F 中的 W_g 与 W_f 后，非残差网络 $G = 1.2$ ，把 G 简化为线性运算 $G(x) = W_g * x$ ，可以明显看出 $W_g = 1.2$ 。残差网络 $H(1) = 1.2$ ， $H(1) = F(1) + 1$ ， $F(1) = 0.2$ ，把 F 简化为线性运算 $F(x) = W_f * x$ ， $W_f = 0.2$

可以明显的看出， F 的参数 W_f 就从 0.1 更新到 0.2，而 G 的参数 W_g 就从 1.1 更新到 1.2，这一点变化对 F 的影响远远大于 G ，说明引入残差后的映射对输出的变化更敏感，对权重的调整作用更大，所以效果更好。

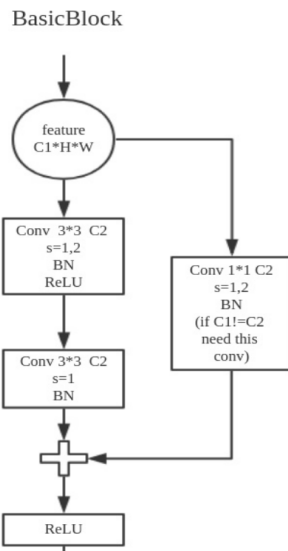
最后，对于残差网络 $H = F(x) + x$ ，每一个导数就加上了一个恒等项 1， $\frac{d_h}{d_x} = \frac{d(f+x)}{d_x} = 1 + \frac{d_f}{d_x}$ ，此时就算原来的导数很小，这时候误差仍然能够有效的反向传播，有效避免了非残差网络链式求导连乘而引发的梯度消散。

C. ResNet-34

该网络除了最开始卷积池化和最后的池化全连接之外，网络中有很多相似的单元，这些重复单元的共同点就是有个跨层直连的 shortcut。

1) **BasicBlock**: ResNet 中使用的一种网络结构，在 resnet18 和 resnet34 中使用了 BasicBlock：输入输出通道数均为 64，残差基础块中两个 3×3 卷积层参数量是： $3 \times 3 \times 64 \times 64 + 3 \times 3 \times 64 \times 64 = 73728$ ，

BasicBlock 类中计算了残差，该类继承了 nn.Module。



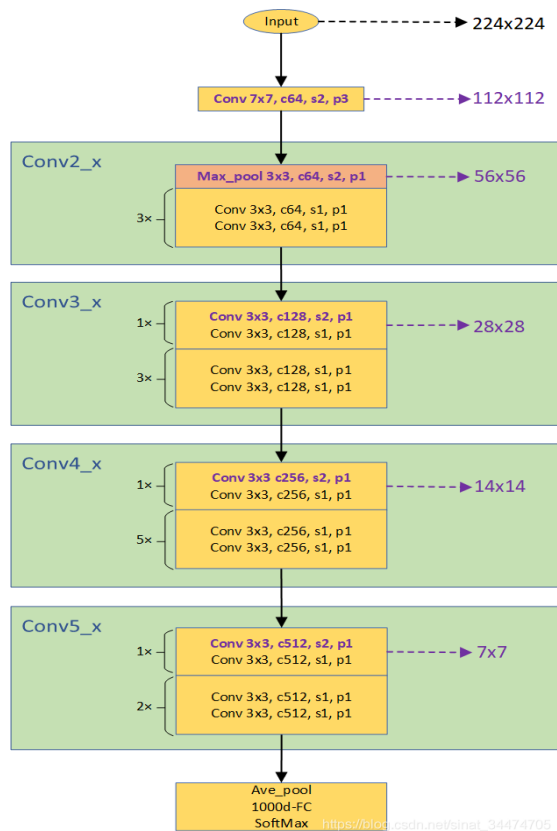


Fig. 4. ResNet-34 结构图

2) ResNet-34 网络结构:

- 每个绿色的矩形框表示 ResNet 的一个 Stage，从上到下为 Stage1 (Conv2_x)、Stage2 (Conv3_x)、Stage3 (Conv4_x)、Stage4 (Conv5_x)
- 绿色矩形框内的每个黄色的矩形框表示 1 个或多个标准的残差单元，黄色矩形框左侧的数值表示残差单元级联的数量，如 5x 表示 5 个级联的残差单元
- 通道数变化：输入通道为 3，4 个 Stage 的通道数依次为 64、128、256、512，即每经过一个 Stage 通道数翻倍
- 层数计算：每个 Stage 包含的残差单元数量依次为 3、4、6、3，每个残差单元包含 2 个卷积层，再算上第一个 7x7 卷积层和 3x3 最大池化层，总的层数 = (3 + 4 + 6 + 3) * 2 + 1 + 1 = 34
- 下采样：黄色矩形框中紫色的部分代表进行下采样操作，即特征图大小减半，右侧箭头标识即为下采样后的特征图大小（输入 224x224 为例）；第一个绿色矩形框内的橙色矩形框表示最大池化，此处发生第一次下采样
- 卷积层参数解释：以 Conv 3x3, c512, s2, p1 为例

3x3 表示卷积核大小，c512 表示卷积核数量/输出通道数量为 512，s2 表示卷积步长为 2，p1 表示卷积的 padding 取 1

- 池化层参数解释：Max_pool 3x3, c64, s2, p1, 3x3 表示池化的区域大小（类似于卷积核大小），c64 表示输入输出通道为 64，s2 表示池化的步长为 2，p1 表示 padding 取 1

III. TRICK 原理介绍

A. 数据增广

数据增广是深度学习中常用的技巧之一，主要用于增加训练数据集，让数据集尽可能的多样化，使得训练的模型具有更强的泛化能力。对于图像分类而言，数据增广的操作有翻转，随机裁剪，随机亮度变化，随机对比度变化，随机旋转等，而标签保持不变。通过数据增强，有可能得到更好的结果。

所以，为了加强模型的泛化能力，我们可以对现有的数据集进行微小的改变。比如旋转 (flips)、移位 (translations)、旋转 (rotations) 等。我们的网络会认为这是不同的图片。一个卷积神经网络，如果能够对物体即使它放在不同的地方也能稳健的分类，就被称为具有不变性的属性。我们通过额外合成的数据来训练神经网络来解释这些情况。

在 tensorflow 中有一些封装好的处理函数，比如在 cifar10_input.py 中就有使用到如下几个：

tf.image.random_crop: 对图像随机裁剪

tf.image.random_flip_left_right: 随机左右翻转

tf.image_random_brightness: 随机亮度变化

tf.image_random_contrast: 随机对比度变化

tf.image_per_image_standardization: 减去均值像素，并除以像素方差（图片标准化）

ToTensor 如果使用的是 Pytorch，则需要将图像转换为 Torch.Tensor。唯一需要注意的是，使用 Pytorch，我们的图像维度中首先是通道，而不是最后是通道。最后，还可以选择张量的输出类型。

1) 旋转: 关于此操作需要注意的一件事是旋转后图像尺寸可能无法保留。如果图像是正方形，则以直角旋转它将保留图像大小。如果它是一个矩形，旋转 180 度将保持大小。以更精细的角度旋转图像也会改变最终的图像尺寸。

2) 缩放: 图像可以向外缩放（放大）或者向内缩放（缩小）。如向外缩放（scaling outward）时，最终图像

尺寸将大于原始图像尺寸，然后大多数图像框架从放大的新图像中剪切出一个部分，其大小等于原始图像。而向内缩放时，它会缩小图像大小，迫使我们超出边界的内容做出假设。

3) 图像切割: 生成比图像尺寸小一些的矩形框，对图像进行随机的切割，最终以矩形框内的图像作为训练数据。

与缩放不同，切割只是从原始图像中随机抽样 (sample) 一个部分。然后，我们将此部分的大小调整为原始图像大小。这种方法通常称为随机裁剪 (random cropping)。数据增广中的缩放与裁剪区别在于 crop 和 resize 的顺序，缩放是先 resize 再 crop，而裁剪时先 crop 再 resize。顺序不同，对生成的图像影响很大，所以缩放和裁剪不能混为一谈。

4) 图像翻转: 可以水平和垂直翻转 (flip) 图像。某些框架不提供垂直翻转功能。但是，垂直翻转相当于将图像旋转 180 度然后执行水平翻转。

5) 图像白化: 白化，又称漂白；是对原始数据 x 实现一种变换，变换成 $x_Whitened$ ，使 $x_Whitened$ 的协方差矩阵的为单位阵。

由于原始图像相邻像素值具有高度相关性，所以图像数据信息冗余，需要做白化处理。

一般情况下，所获得的数据都具有相关性，所以通常都要求对数据进行初步的白化或球化处理，因为白化处理可去除各观测信号之间的相关性，从而简化了后续独立分量的提取过程，而且，通常情况下，数据进行白化处理与不对数据进行白化处理相比，算法的收敛性较好。

白化这种常规的方法作为 ICA 的预处理可以有效地降低问题的复杂度，而且算法简单，用传统的 PCA 就可完成。用 PCA 对观测信号进行白化的预处理使得原来所求的解混合矩阵退化成一个正交阵，减少了 ICA 的工作量。此外，PCA 本身具有降维功能，当观测信号的个数大于源信号个数时，经过白化可以自动将观测信号数目降到与源信号维数相同。

白化的作用的描述主要有两个方面：1，减少特征之间的相关性；2，特征具有相同的方差（协方差阵为 1）。

6) 模糊: 有很多方法可以随机模糊图像。最著名的是均值，中值，高斯和双边滤波器。

平均模糊

关于平均滤波器，顾名思义，它使我们可以对给定中心的值取平均值。这是由内核完成的。可以指定其大

小以增加或减少模糊。要使用平均滤波器增广图像，我们只需要使用随机大小的内核对输入图像进行滤波。

高斯模糊

高斯模糊不使用平均滤波器，而是使用高斯滤波器，因此这些值对应于从中心开始的高斯曲线。注意内核维数只能包含奇数。

7) 色彩空间: 如果我们知道色彩空间，则可以利用它们的属性来增广图像。举一个简单的例子，借助 HSV 颜色空间，我们可以很容易地提取树叶的颜色，并根据我们的意愿随机更改其颜色。并且我们可以了解自己的图像增广功能的原理。因此，重要的是要了解色彩空间，以充分利用它们。

8) 数据增强: 几种数据增强的比较：

- Mixup: 将随机的两张样本按比例混合，分类的结果按比例分配；
- Cutout: 随机的将样本中的部分区域 cut 掉，并且填充 0 像素值，分类的结果不变；
- CutMix: 就是将一部分区域 cut 掉但不填充 0 像素而是随机填充训练集中的其他数据的区域像素值，分类结果按一定的比例分配。

上述三种数据增强的区别：cutout 和 cutmix 就是填充区域像素值的区别；mixup 和 cutmix 是混合两种样本方式上的区别：mixup 是将两张图按比例进行插值来混合样本，cutmix 是采用 cut 部分区域再补丁的形式去混合图像，不会有图像混合后不自然的情形。这些





	ResNet-50	Mixup [48]	Cutout [3]	CutMix
Image				
Label	Dog 1.0	Dog 0.5 Cat 0.5	Dog 1.0	Dog 0.6 Cat 0.4

Fig. 5. 数据增强

数据增强方法的优点是：

- 1) 在训练过程中不会出现非信息像素，从而能够提高训练效率；
- 2) 保留了 regional dropout 的优势，能够关注目标的 non-discriminative parts；
- 3) 通过要求模型从局部视图识别对象，对 cut 区域中添加其他样本的信息，能够进一步增强模型的定位能力；

- 4) 不会有图像混合后不自然的情形，能够提升模型分类的表现；
- 5) 训练和推理代价保持不变。

B. 正则化

1) 批次正则化 (Batch Normalization) : 在反向传播过程中，每层权重的更新是在假定其他权重不变的情况下，向损失函数降低的方向调整自己。而问题在于，在一次反向传播过程中，所有的权重会同时更新，导致层间配合“缺乏默契”，下一次更新时又要根据新的分布重新调整，而且层数越多，相互配合越困难。为了避免过于震荡，学习率不得不设置得足够小，这就意味着学习速度缓慢。为此，希望对每层输入的分布有所控制，于是就有了 Batch Normalization，其出发点是对每层的输入做 Normalization。Batch Normalization 是神经网络中一种特殊的层，如今已是各种流行网络的标配。通常 BN 被建议插入在 ReLU 激活层前面，如图所示。

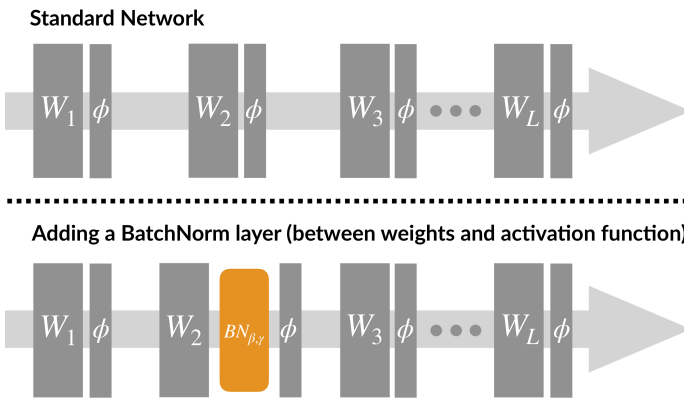


Fig. 6. 插入 BN 层的神经网络结构

如果 batch size 为 m ，则在前向传播过程中，网络中每个节点都有 m 个输出，所谓的 Batch Normalization，就是对该层每个节点的这 m 个输出进行正则化再输出。在图像预处理过程中通常会图像进行标准化处理，这样能够加速网络的收敛，如图 7 所示，对于 Conv1 来说输入的就是满足某一分布的特征矩阵，但对于 Conv2 而言输入的 feature map 则不一定满足某一分布规律，这里所说满足某一分布规律并不是指某一个 feature map 的数据要满足分布规律，理论上是指整个训练样本集所对应 feature map 的数据要满足分布规律。Batch Normalization 的目的就是使神经网络 feature map 满足均值为 0，方差为 1 的分布规律。

对于一个拥有 d 维的输入 x ，我们将对它的每一个维度进行标准化处理。假设我们输入的 x 是 RGB 三通

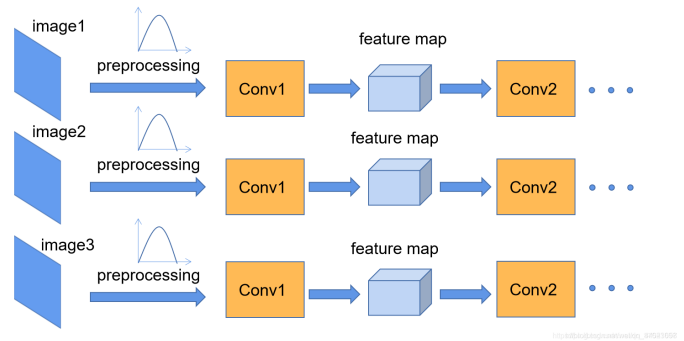


Fig. 7. 图像进行标准化处理

道的彩色图像，那么这里的 d 就是输入图像的 channels 即 $d=3$ ，其中就代表我们的 R 通道所对应的特征矩阵，依此类推。图像标准化处理也就是分别对我们的 R 通道，G 通道，B 通道进行处理。假设 BN 层有 d 个输入节点，则 x 可构成 $d \times m$ 大小的矩阵 X ，BN 层相当于通过行操作将其映射为另一个 $d \times m$ 大小的矩阵 Y ，如图所示，将 2 个过程写在一个公式里如下：

$$y_i^{(b)} = B * N * (x_i)^{(b)} = \gamma * \frac{x_i^{(b)} - (x_i)}{\sqrt{\sigma(x_i)^{(b)} + \epsilon}}$$

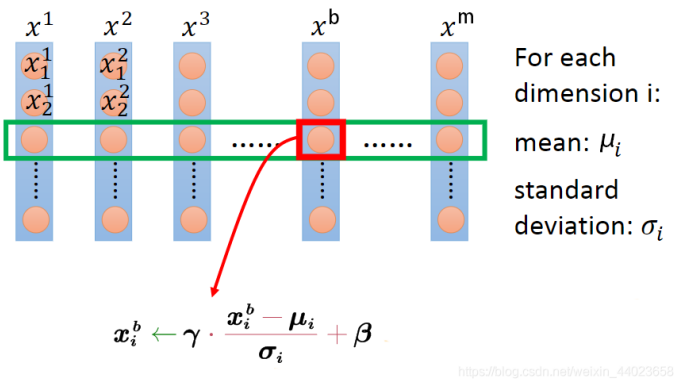


Fig. 8. BN 层功能

其中， $x_i^{(b)}$ 表示输入当前 batch 的 b -th 样本时该层 i -th 输入节点的值， x_i 为 $[x(1)_i, x(2)_i, \dots, x(m)_i]$ 构成的行向量，长度为 batch size m ， μ 和 σ 为该行的均值和标准差， ϵ 为防止除零引入的极小量（可忽略）， γ 和 β 为该行的 scale 和 shift 参数，可知

- μ 和 σ 为当前行的统计量，不可学习。
- γ 和 β 为待学习的 scale 和 shift 参数，用于控制 y_i 的方差和均值
- BN 层中， x_i 和 x_j 之间不存在信息交流 ($i \neq j$)

可见, 无论 x_i 原本的均值和方差是多少, 通过 Batch-Norm 后其均值和方差分别变为待学习的 μ 和 γ 。

使用 Batch Normalization, 可以获得如下好处,

- 可以使用更大的学习率, 训练过程更加稳定, 极大提高了训练速度。
- 可以将 bias 置为 0, 因为 Batch Normalization 的 Standardization 过程会移除直流分量, 所以不再需要 bias。
- 对权重初始化不再敏感, 通常权重采样自 0 均值某方差的高斯分布, 以往对高斯分布的方差设置十分重要, 有了 Batch Normalization 后, 对与同一个输出节点相连的权重进行放缩, 其标准差 σ 也会放缩同样的倍数, 相除抵消。
- 对权重的尺度不再敏感, 理由同上, 尺度统一由 γ 参数控制, 在训练中决定。
- 深层网络可以使用 sigmoid 和 tanh 了, 理由同上, BN 抑制了梯度消失。
- Batch Normalization 具有某种正则作用, 不需要太依赖 dropout, 减少过拟合。

2) 通道正则化 (Group Normalization) : BN 是在一个 batch 中计算均值和方差, BN 可以简化并优化使得非常深的网络能够收敛。但是 BN 却很受 batch 大小的影响, 通过实验证明: BN 需要一个足够大的批量, 小的批量大小会导致对批统计数据的不准确率提高, 显著增加模型的错误率。比如在检测、分割、视频识别等任务中, 比如在 faster R-cnn 或 mask R-cnn 框架中使用一个 batchsize 为 1 或 2 的图像, 因为分辨率更高, 其中 BN 被“冻结”转换为线性层。

为了解决 BN 对较小的 mini-batch size 效果差的问题, GN 应运而生。GN 适用于占用显存比较大的任务, 例如图像分割。对这类任务, batch size 可能只能是个位数, 再大显存就不够用了。而当 batch size 是个位数时, BN 的表现很差, 因为没办法通过几个样本的数据量, 来近似总体的均值和标准差

GN 在各种批量大小上表现得非常稳定。对于批量大小为 2 的样本, GN 在 ImageNet 上的 ResNet-50 中误差率比 BN 对应的低 10.6%。对于一般的批量大小, GN 性能与 BN 相当 (差距约为 0.5%), 并且优于其他归一化变体。此外, 尽管批量大小可能会发生变化, 但 GN 可以自然地由预训练转移到微调。对于 Mask R-CNN 上的 COCO 目标检测和分割, 以及用于 Kinetics 视频分类的 3D 卷积网络, GN 显示了与 BN

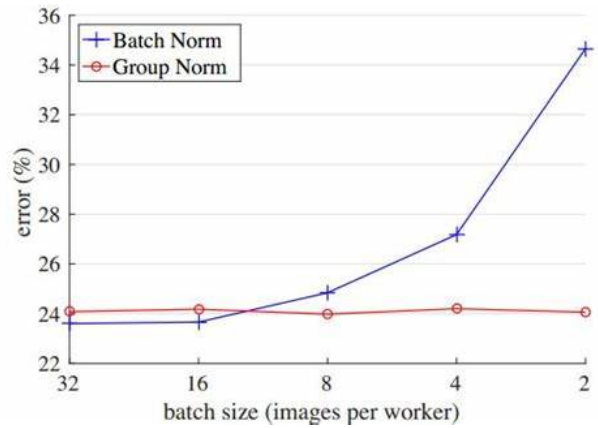


Fig. 9. BN 与 GN 对于较小的 batch size 的不同表现

对应物相比改进的结果。GN 在 ImageNet, COCO 和 Kinetics 中的有效性表明, GN 是 BN 的竞争替代品, 在这些任务中占主导地位。

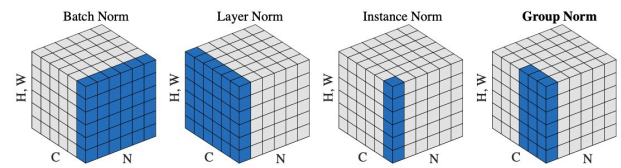


Fig. 10. 四种正则化的工作方式

对于 GN(Group Normalization) 的操作如图 11 所示, 假设 num_groups=2, 原论文中默认为 32, 由于和 batch_size 无关, 我们直接看对于一个样本的情况。假设某层输出得到 x , 根据 num_groups 沿 channel 方向均分成 num_groups 份, 然后对每一份求均值和方差, 接着按照下面的公式进行计算即可。

$$y = \frac{x - E(x)}{\sqrt{\text{Var}(x) + \epsilon}} * \gamma + \beta$$

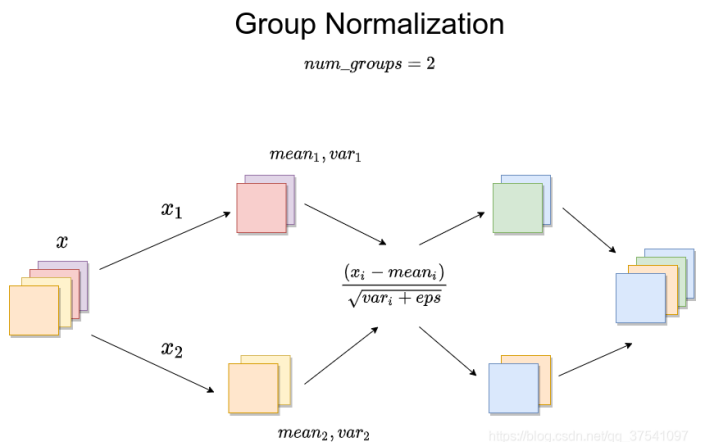


Fig. 11. GN 层操作

C. 梯度裁剪

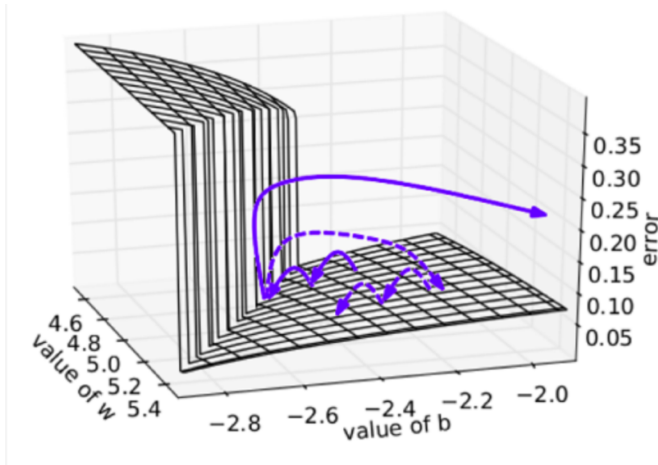


Fig. 12. 梯度爆炸

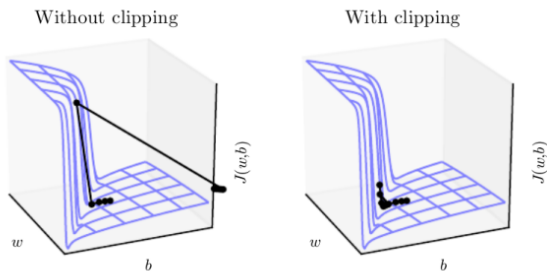


Fig. 13. 梯度裁剪

深度神经网络采用链式法则进行参数求导的方式并不是绝对安全的，有时会出现梯度消失或者梯度爆炸的情况。其中梯度爆炸问题是指训练过程中梯度范数的大幅增加（如图 12 所示）。梯度爆炸问题一种常见的方法是 L2 正则化，它在网络的成本函数中应用“权重衰减”。正则化参数变大，权重变小，有效地降低了参数占比，从而使模型更加线性。但是梯度剪裁技术在实施和结果获取方面有更好的优越性。

梯度裁剪 (Gradient Clipping)，也就是通过“clip”方式来防止迭代中梯度值过大，它是一种在网络反向传播期间将误差导数更改或裁剪到阈值的方法，并使用裁剪后的梯度来更新权重。通过重新调整误差导数，对权重的更新也将重新调整，从而显著降低上溢或下溢的可能性（如图 15 所示）。pytorch 中使用 `nn.utils.clip_grad_norm_()` 进行梯度裁剪，该函数有三个参数：

- `parameters` —— 一个基于变量的迭代器，会进行梯度归一化

- `max_norm` —— 梯度的最大范数
- `norm_type` —— 规定范数的类型，默认为 L2

TensorFlow 里提供了一系列简单可行的梯度裁剪函数，方便我们对超过阈值的梯度值进行规约，使优化算法相对更加数值稳定：这几个梯度裁剪函数都是以 `clip_by` 开头，分别是 `tf.clip_by_norm`，`tf.clip_by_global_norm`，`tf.clip_by_average_norm` 和 `tf.clip_by_value`。

D. dropout

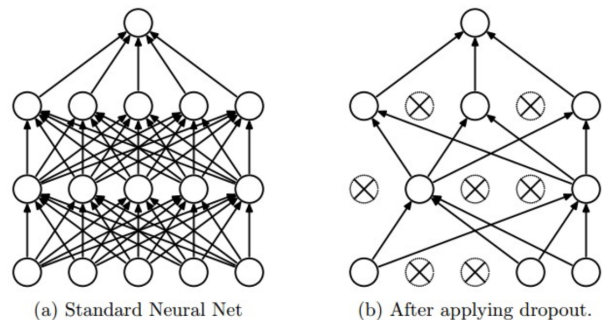


Fig. 14. dropout

在机器学习的模型中，如果模型的参数太多，而训练样本又太少，训练出来的模型很容易产生过拟合的现象。在训练神经网络的时候经常会遇到过拟合的问题，过拟合具体表现在：模型在训练数据上损失函数较小，预测准确率较高；但是在测试数据上损失函数比较大，预测准确率较低。为了解决过拟合问题，一般会采用模型集成的方法，即训练多个模型进行组合。此时，训练模型费时就成为一个很大的问题，不仅训练多个模型费时，测试多个模型也是很费时。

在这种问题的背景下，用于避免过拟合的 dropout 技术应运而生，在 2012 年，Hinton 在其论文《Improving neural networks by preventing co-adaptation of feature detectors》中提出 Dropout。当一个复杂的前馈神经网络被训练在小的数据集时，容易造成过拟合。为了防止过拟合，可以通过阻止特征检测器的共同作用来提高神经网络的性能。同年，Alex、Hinton 在其论文《ImageNet Classification with Deep Convolutional Neural Networks》中用到了 Dropout 算法，一举赢得了 2012 年图像识别大赛冠军，而这篇论文提到的 AlexNet 网络模型也引爆了神经网络应用热潮。

dropout 在神经网络的训练过程中，对于一次迭代中的某一层神经网络，先随机选择其中的一些神经元并

将其临时隐藏 (丢弃), 然后再进行本次训练和优化。在下次迭代中, 继续随机隐藏一些神经元, 如此直至训练结束。由于是随机丢弃, 故而每一个 mini-batch 都在训练不同的网络。

在训练时, 每个神经单元以概率 p 被保留 (Dropout 丢弃率为 $1-p$); 在预测阶段 (测试阶段), 每个神经单元都是存在的, 权重参数 w 要乘以 p , 输出是: pw (如图 14 所示)

E. 权重衰减

L_2 正则化的目的就是为了让权重衰减到更小的值, 在一定程度上减少模型过拟合的问题, 所以权重衰减也叫 L_2 正则化。

1) L_2 正则化与权重衰减系数: L_2 正则化就是在代价函数后面再加上一个正则化项:

$$C = C_0 + \frac{\lambda}{2n} \sum_w w^2$$

其中 C_0 代表原始的代价函数, 后面那一项就是 L_2 正则化项, 它是这样来的: 所有参数 w 的平方的和, 除以训练集的样本大小 n 。就是正则项系数, 权衡正则项与 C_0 项的比重。另外还有一个系数 $\frac{1}{2}$, $\frac{1}{2}$ 经常会看到, 主要是为了后面求导的结果方便, 后面那一项求导会产生一个 2, 与 $\frac{1}{2}$ 相乘刚好凑整为 1。系数 λ 就是权重衰减系数。

2) 为什么可以对权重衰减: 我们对加入 L_2 正则化后的代价函数进行推导, 先求导:

$$\frac{\partial C}{\partial w} = \frac{\partial C_0}{\partial w} + \frac{\lambda}{n} w \frac{\partial C}{\partial w} = \frac{\partial C_0}{\partial w}$$

可以发现 L_2 正则化项对 b 的更新没有影响, 但是对于 w 的更新有影响:

$$\begin{aligned} w &\rightarrow w - \eta \frac{\partial C_0}{\partial w} - \frac{\eta \lambda}{n} w \\ &= \left(1 - \frac{\eta \lambda}{n}\right) w - \eta \frac{\partial C_0}{\partial w} \end{aligned}$$

在不使用 L_2 正则化时, 求导结果中 w 前系数为 1, 现在 w 前面系数为 $\frac{1-\eta\lambda}{n}$, 因为 η, λ, n 都是正的, 所以 $\frac{1-\eta\lambda}{n}$ 小于 1, 它的效果是减小 w , 这也就是权重衰减 (weight decay) 的由来。当然考虑到后面的导数项, w 最终的值可能增大也可能减小。另外, 需要提一下, 对于基于 mini-batch 的随机梯度下降, w 和 b 更新的公式跟上面给出的有点不同:

$$\begin{aligned} w &\rightarrow \left(1 - \frac{\eta \lambda}{n}\right) w - \frac{\eta}{m} \sum_x \frac{\partial C_x}{\partial w} \\ b &\rightarrow b - \frac{\eta}{m} \sum_x \frac{\partial C_x}{\partial b} \end{aligned}$$

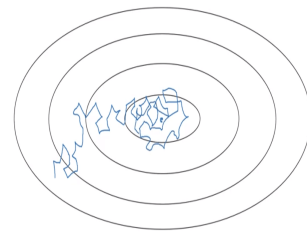
对比上面 w 的更新公式, 可以发现后面那一项变了, 变成所有导数加和, 乘以 η 再除以 m , m 是一个 mini-batch 中样本的个数。

3) 权重衰减 (L_2 正则化) 的作用:

- 作用: 权重衰减 (L_2 正则化) 可以避免模型过拟合问题。
- 思考: L_2 正则化项有让 w 变小的效果, 但是为什么 w 变小可以防止过拟合呢?
- 原理:
 - 1) 从模型的复杂度上解释: 更小的权值 w , 从某种意义上说, 表示网络的复杂度更低, 对数据的拟合更好 (这个法则也叫做奥卡姆剃刀), 而在实际应用中, 也验证了这一点, L_2 正则化的效果往往好于未经正则化的效果。
 - 2) 从数学方面的解释: 过拟合的时候, 拟合函数的系数往往非常大, 为什么? 如图所示, 过拟合, 就是拟合函数需要顾忌每一个点, 最终形成的拟合函数波动很大。在某些很小的区间里, 函数值的变化很剧烈。这就意味着函数在某些小区间里的导数值 (绝对值) 非常大, 由于自变量值可大可小, 所以只有系数足够大, 才能保证导数值很大。而正则化是通过约束参数的范数使其不要太大, 所以可以在一定程度上减少过拟合情况。

F. 学习率衰减

1) 为什么需要学习率衰减: mini-batch 梯度下降算法可以提高更新权重的速度, 让我们及时看到损失函数的情况, 但是每个损失函数并不会一直下降, 而是在保证整体趋势减小的情况下略微波动, 如果用一个等高线图来表示就是这样的:



下图 III-F.1 的中心点为最优值点, 我们可以看到损失函数渐渐接近最优值, 但最后却在最优值附近

摆动，这是因为学习率的大小一直不变，如果我们可以让学习率随着损失函数接近最优值或是随着迭代次数的增加而慢慢减小的话，就可以得到图中的绿线：



可以看到，随着损失函数接近最优值，摆动的幅度也在减小，从而保证最后损失函数在离最优值更近的范围内摆动。

2) 实现：我们的目标是让学习率 α ，随着迭代次数的增加而逐渐减小。先明确迭代这个概念，所谓迭代就是让训练集中的所有数据都输入一次网络，这就是一次迭代。如果训练集被分成了许多个 mini batches，那一次迭代就是所有 mini batches 中的数据都依次输入进网络并更新了权重。因此我们可以这样定义学习率：

$$\alpha = \frac{1}{1 + \text{decayRate} \cdot \text{epochNum}} \cdot \alpha_0$$

其中，

- decayRate 是衰减率，衰减率越大，学习率就减小得越快
- epochNum 是迭代数
- α_0 是初始学习率，初始学习率可以设置得大一些，这样可以保证算法一开始的学习速度不会太慢

从这个式子可以明显看出，随着迭代次数 epochNum 的增大， α 就会变小。

decayRate 和 α_0 都是超参数，我们需要在实践中调整这些值得到满意的结果。

IV. 类别不均衡问题

A. 什么是类别不平衡

类别不平衡 (class-imbalance)，也叫数据倾斜，数据不平衡，就是指分类任务中不同类别的训练样例数目差别很大的情况。传统的分类算法旨在最小化分类过程中产生的错误数量。它们假设假阳性（实际是反例，但是错分成正例）和假阴性（实际是正例，但是错分为反

例）错误的成本是相等的，因此不适合于类不平衡的数据。

有研究表明，在某些应用下，1 : 35 的比例就会使某些分类方法无效，甚至 1 : 10 的比例也会使某些分类方法无效。如果数据存在严重的不平衡，预测得出的结论往往也是有偏的，即分类结果会偏向于较多观测的类。

标准机器学习算法通常假设不同类别的样本数量大致相似，所以类别不平衡现象会导致学习算法效果大打折扣。因此有必要了解类别不平衡时处理的基本方法。在 cifar10 分类学习任务中，我们研究了遇到类别不平衡问题时的解决方案。

B. 提升不平衡类分类准确率的方法

提升不平衡类分类准确率的方法有三大类：采样、阈值移动、调整代价或权重。



Fig. 15. 不平衡类分类准确率的方法

1) 过采样：过采样基本思想就是通过改变训练数据的分布来消除或减小数据的不平衡。过采样有随机过采样、基于聚类的过采样、信息性过采样（SMOTE）三大类方法。

- 1) 随机过采样：通过增加少数类样本来提高少数类的分类性能，最简单的办法是随机复制少数类样本。
- 2) 基于聚类的过采样：K-Means 聚类算法独立地被用于少数和多数类实例，之后，每个聚类都过采样使得相同类的所有聚类有着同样的实例数量。
- 3) 信息性过采样-SMOTE：利用 KNN 技术，对于少数类样本 a，随机选择一个最近邻的样本 b，然后从 a 与 b 的连线上随机选取一个点 c 作为新的少数类样本。SMOTE 有时能提升分类准确率，

有时不能，甚至可能因为构建数据时放大了噪声数据导致分类结果变差，这要视具体情况而定。

2) **欠采样**: 欠采样方法通过减少多数类样本来提高少数类的分类性能。

- 1) **随机欠采样**: 通过随机地去掉一些多数类样本来减小多数类的规模。
- 2) **集成技术**: 欠采样中的算法集成技术是利用集成学习机制，将反例划分为若干个集合供不同学习器使用，这样对每个学习器来看都进行了欠采样，但在全局来看却不会丢失重要信息，一般适用于数据集足够大的情况。这里的集成技术可以分为基于 Bagging 的方法和基于 Boosting 的方法。

3) **阈值移动法**: 许多模型的输出类别是基于阈值的，例如逻辑回归中小于 0.5 的为反例，大于则为正例。在数据不平衡时，默认的阈值会导致模型输出倾向于类别数据多的类别。阈值移动是通过改变决策阈值来偏重少数类。

4) **调整代价或权重法**: 通过调整不同类的代价或权重来偏重少数类以改进分类性能。

V. 部分标注问题

A. 为什么有部分标注

得益于深度学习的发展，许多计算机视觉任务在近几年取得了不错的效果。但是，现有的深度学习算法多是有监督学习算法，依赖大量人工标记的训练数据，而标注数据十分耗费人力成本。因此，解决深度学习对数据的依赖问题和减少数据标注成本成为了业界的 research 热点。

B. 半监督/弱监督学习

半监督学习是监督学习和无监督学习相结合的一种学习方法。半监督/弱监督学习使用大量的未标注数据/弱标注数据，同时使用小部分已标注数据，来训练机器学习模型。它预期的结果是通过利用大部分未标注数据/弱标注数据的利用，得到的模型优于单纯只用已标注数据训练的模型。弱标注数据的数据标签信息量较少且标注难度小，比如在目标检测任务中，通常需要标注目标的类别和坐标，弱标注数据则只标注出图像中的目标类别，没有坐标信息。

C. 数据合成

既然有监督学习无法避免模型对标注数据的依赖，那么自动生成数据也是减少人工成本的一个方式。数据合成的方式很多，包括人工设计规则，使用 GAN 网络生成等。以文本识别为例，可以采取以下基于人工设计规则的合成数据方法。合成的图像样本由前景图像层、背景图像层、边缘/阴影层组成，合成步骤分为六步：

- font rendering: 随机选择字体并将文本呈现入前景层；
- border/shadow rendering: 从前景层的文字中产生边缘阴影等；
- base coloring: 给三个图层填色；
- projective distortion: 对前景和阴影进行随机扭曲变换；
- natural data blending: 将图像跟真实场景图像进行混合；
- noise: 加入高斯噪声等。

D. 主动学习

不同样本对现有模型的提升帮助是不同的，正如人类的学习过程一样，只学习小学知识的人很难突破初中知识的瓶颈。主动学习的出发点与此类似，就是希望从未标注数据集中挑选对模型提升帮助最大的子集交给人工标注。因此在标注同样数据量的样本的情况下（同样的标注成本），采用主动学习策略挑选样本训练的模型是接近最优的。主动学习的流程如图所示，左侧的已标注数据集训练得到模型，模型在未标注数据集上推理，并将标注意义较大的样本推给人工标注，再将新标注的数据集重新训练和提升模型。

E. 自监督

自监督学习是无监督学习的一种，近期是学术界的研究热点。它通过利用无标签的数据本身的结构或者特性，人为构造标签出来监督网络学习。通常自监督学习的模型并不直接应用在目标任务上，而是作为下游任务的预训练模型。在自监督学习的最新进展中，使用该方法得到的无监督模型，作为预训练模型在许多下游任务 fine-tune 后的效果优于使用有监督学习的预训练模型。

VI. 延伸与拓展

这个部分是我们在 cifar10 任务上的一些深入研究与思考，相较于调参和数据增强，模型的优化常常能带

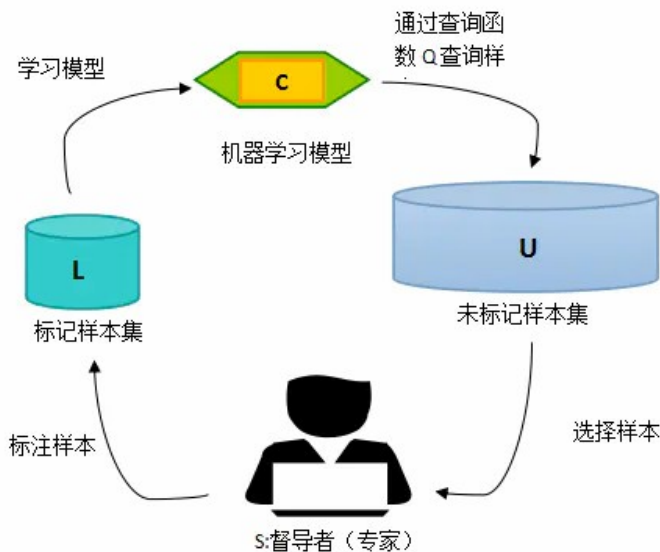


Fig. 16. 主动学习的流程图

来更多的提升，故我们复现了该数据集的一个 sota；然而，这并不能否认调参优化的重要性，我们同时也尝试了目前一个比较热门的领域 autoML，以经典模型的自动调参，同样也取得了非常优异的结果。

A. Going deeper with Image Transformers

在计算机视觉领域中，多数算法都是保持 CNN 整体结构不变，在 CNN 中增加 attention 模块或者使用 attention 模块替换 CNN 中的某些部分。有研究者提出，没有必要总是依赖于 CNN。因此，作者提出 ViT 算法，仅仅使用 Transformer 结构也能够在图像分类任务中表现很好。

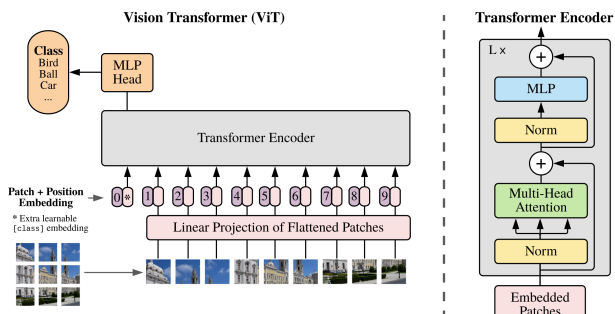


Fig. 17. ViT 算法结构示意图

受到 NLP 领域中 Transformer 成功应用的启发，ViT 算法中尝试将标准的 Transformer 结构直接应用于图像，并对整个图像分类流程进行最少的修改。具体来讲，ViT 算法中，会将整幅图像拆分成小图像块，

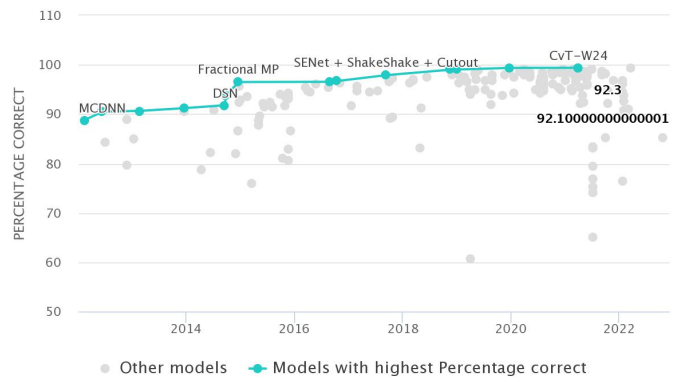


Fig. 18. sota 排行榜

然后把这些小图像块的线性嵌入序列作为 Transformer 的输入送入网络，然后使用监督学习的方式进行图像分类的训练。

Transformer 被用于大规模的图像分类后，取得了高分，动摇了卷积神经网络长期以来的优势地位。然而，到目前为止，对图像 Transformer 的优化研究还很少。在 sota 中，作者为图像分类建立并优化了更深层次的 Transformer 网络。特别是，作者研究了这种专用转化器的架构和优化的相互作用。他们做了两个 Transformer 架构的改变，大大改善了深层 Transformer 的准确性。这使产生的模型的性能不会随着深度的增加而提前饱和。

最终该 sota 取得了 99.4% 的准确率。

B. AutoML

机器学习的应用需要大量的人工干预，这些人工干预表现在：特征提取、模型选择、参数调节等机器学习的各个方面。AutoML 视图将这些与特征、模型、优化、评价有关的重要步骤进行自动化地学习，使得机器学习模型无需人工干预即可被应用。

这是自动化机器学习 (AutoML) 背后的理念，也是 AutoGluon AutoML 库的设计理念，该库由 Amazon Web Services (AWS) 在 re:invent 2019 上开源。使用 AutoGluon，可以训练最先进的机器学习模型，以进行图像分类、对象检测、文本分类和表格式数据预测，而几乎无需具备机器学习方面的经验。

将多个模型组合起来，创建一个“集成”，从而相比每个参与者都具有更高的预测准确度，这种想法并不新鲜。集成技术最早的实现可以追溯到 20 世纪 90 年代早期，当时出现了提升法（和 AdaBoost 算法）和自助聚合 (Bootstrap Aggregation)。这些技术创建了决策树



Fig. 19. AutoML

的集成, 这些决策树是弱分类器 (不比随机猜测强多少) 且不稳定 (对数据集的变化很敏感)。但是, 当许多决策树组合在一起时, 就生成了具有高预测能力的模型, 不受过拟合的影响。这些早期成果是许多流行机器学习包的基础, 如 LightGBM、CatBoost 和 scikit-learn 的 RandomForest, 这些都在 AutoGluon 中所采用。

从机器学习角度讲, AutoML 可以看作是一个在给定数据和任务上学习和泛化能力非常强大的系统。但是它强调必须非常容易使用。- 从自动化角度讲, AutoML 则可以看作是设计一系列高级的控制系统去操作机器学习模型, 使得模型可以自动化地学习到合适的参数和配置而无需人工干预。最终我们利用该框架, 仅用五行代码便达到了 96.2% 的准确率。

VII. 实验分析与对比

在没有学习相关知识之前, 我们以为唯有模型的不断加深, 架构的不断优化, 才能提高模型的准确率。但经过本次作业的研究与学习之后, 我们发现, 提高准确率的技巧多种多样, 科研人员的探索远远超乎我们的想象。

具体的代码与实现方式在报告中不再赘述, 详见附件。本次作业的目的是体会各种各样优化模型方法的效果, 分析其中的优劣, 而不是追求图像分类的准确率。故训练集只随机取了 10000 张, 且模型从 Lenet, Alexnet, 到最终选定 ResNet 之后就不再做大的改动, 以此为 baseline 进行实验。全篇报告论述下的准确率汇总见表格所示。

可以看出随着方法的增加或精进, 模型的准确率也在不断提高。值得注意的是文章中提到的批次正则化、学习率衰减等结合在我的 baseline 里, 故没有单独列出。图像增强的方法选用了随机裁剪与随机旋转。另外通过查阅资料得知, BN 在训练过程对每个单个样本的 forward 均引入多个样本 (Batch 个) 的统计信息, 相当于自带一定噪音, 起到正则效果, 所以也就基本消除了 Dropout 的必要。

如果只对图像做一种增强时, 采用的变换有可能会

表 1 准确率汇总

类型	pytorch	类型	tensorflow
ResNet34_BN	45%	ResNet34	49.93%
CaiT-M36	99.4%	白化/亮度/对比度	50.28%
Autogluon	96.2%	Dropout	63%
随机裁剪/翻转	46%	LRN	72%
Cutout	47%	变化学习率	66%
Mixup	49%	Mixup	52.02%
Cutmix	53%	Clip_gradient	51.47%
ResNet34_GN	59%	Weight_decay	48.88%
Clip_grad	57%	AutoAugment	67%
Weight_decay	48%	Gridmask	67%
类别不平衡	46%	Random-augment	67%
部分标注	89%	HideAndSeek	57%

使图像的关键信息丢失, 从而导致后续任务得不到正确的结果。通过分析多项数据增强的图像, 然后综合分析, 有可能会平滑掉某一种变换导致的关键信息丢失现象带来的损失, 从而提升预测的准确率。我想这也是表格的准确率呈现上升趋势的原因。

在使用数据增广后, 由于训练数据更难, 所以训练损失函数可能较大, 训练集的准确率相对较低, 但其拥有更好的泛化能力, 所以验证集的准确率相对较高。在使用数据增广后, 模型可能会趋于欠拟合状态, 故可以适当的调小 `l2_decay` 的值来获得更高的验证集准确率。

为了实现随机抽取 10000 张的功能我额外编写了一个数据生成脚本, 通过该脚本指定类别数量为 500-1000 的随机数, 导致了类别不平衡的情况。由此我创建了一个共 7527 张图片的训练集 (各类别的样本量未知), 按照 baseline 的方法进行了第一次训练, 准确率 26%, 符合预期。然后我使用了 WeightedRandomSampler, 将图片按照类别的样本量进行随机采样, 成功将准确率提升到 46%。

最后关于部分标注问题我们计划使用半监督学习来解决。通过阅读文献我们得知, 半监督学习 (Semi-Supervised Learning, SSL) 的 SOTA 一次次被 Google 刷新, 从 MixMatch 开始, 到同期的 UDA、ReMixMatch, 再到 2020 年的 FixMatch。在代码中

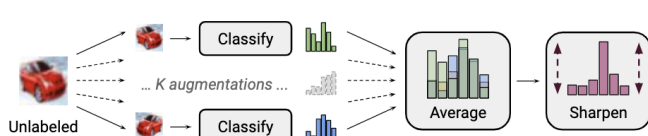


Fig. 20. MixMatch

我们尝试复现了 MixMatch, 该模型集成了自洽正则化, 数据增广使用了对图像的随机左右翻转和剪切, 且使用“sharpening”函数, 来最小化未标记数据的熵, 训练效果优异。官方文档中使用的是 tensorflow, 我们下载了一个开源非官方的 pytorch 版本并修复了版本差异导致的错误。最终使用 250 张标注数据和 44750 张无标注数据进行训练, 取得了 89% 的准确率。

VIII. 总结

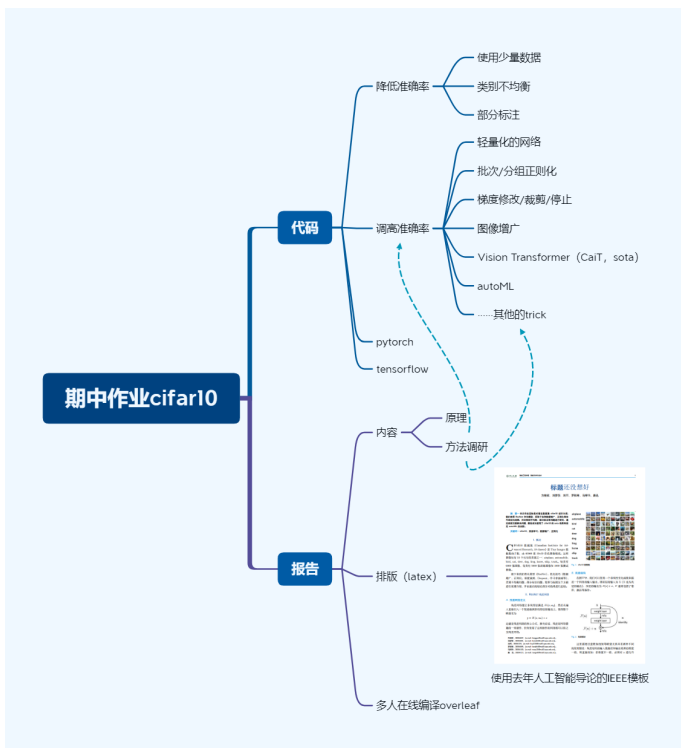


Fig. 21. 完成思路

这张图是我们小组第一次开会时讨论出来的思维导图, 也是我们完成该项目的思路。其中方同学负责了 pytorch 部分的代码实现, 马同学负责了 tensorflow 部分的代码实现, 其他四位同学共同完成调研与报告的编写。

完成该作业的过程中我们深刻体会到前沿科研人员在深度学习领域的探索之深之广, 并不是简简单单搭建一个网络就能实现图像分类等任务, 数据处理、调参、

优化等都是深度学习中深奥的学问, 本次作业里我们接触到的也仅仅是冰山一角。

通过本次实验, 我们了解了 tensorflow 和 pytorch 的原理, 分析了各种各样方法对分类性能的影响, 在实践中感受到深度学习的魅力。

参考文献

- [1] Krizhevsky A, Hinton G. Learning multiple layers of features from tiny images[J]. Handbook of Systemic Autoimmune Diseases, 2009, 1(4).
- [2] He K, Zhang X, Ren S, et al. Deep residual learning for image recognition[C]//Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR). [S.l.: s.n.], 2016.
- [3] Devries T, Taylor G W. Improved Regularization of Convolutional Neural Networks with Cutout[J]. 2017.
- [4] Zhang H, Cisse M, Dauphin Y N, et al. mixup: Beyond Empirical Risk Minimization[J]. 2017.
- [5] Yun S, Han D, Chun S, et al. CutMix: Regularization Strategy to Train Strong Classifiers With Localizable Features[C]// International Conference on Computer Vision. 0.
- [6] Taylor L, Nitschke G. Improving Deep Learning using Generic Data Augmentation[J]. 2017.
- [7] Wu S, Li G, Deng L, et al. L1-Norm Batch Normalization for Efficient Training of Deep Neural Networks[J]. IEEE Transactions on Neural Networks & Learning Systems, 2018:1-9.
- [8] Wu Y, He K. Group Normalization[J]. International Journal of Computer Vision, 2018.
- [9] Hinton G E, Srivastava N, Krizhevsky A, et al. Improving neural networks by preventing co-adaptation of feature detectors[J]. Computer Science, 2012, 3(4):págs. 212-223.
- [10] Loshchilov I, Hutter F. Decoupled Weight Decay Regularization[J]. 2017.
- [11] Technicolor T, Related S, Technicolor T, et al. ImageNet Classification with Deep Convolutional Neural Networks [50].
- [12] J Zhang, He T, Sra S, et al. Why gradient clipping accelerates training: A theoretical justification for adaptivity[J]. 2019.
- [13] Berthelot D, Carlini N, Goodfellow I, et al. Mixmatch: A holistic approach to semi-supervised learning[J]. arXiv preprint arXiv:1905.02249, 2019.
- [14] Klein A, Tiao L C, Lienart T, et al. Model-based Asynchronous Hyperparameter and Neural Architecture Search[J]. 2020.
- [15] Touvron H, Cord M, Sablayrolles A, et al. Going deeper with Image Transformers[J]. 2021.