Deep Learning Homework 02：不平衡数据集上模型训练的改进

520030910155 邱一航

1. 在 Cifar-10 数据集上的训练

笔者搭建的网络结构如下：（对卷积层，其他参数为 padding 模式；对池化层，其他参数为池化层性质（Mean/Max Pooling））

id	性质	核尺寸 × 通道数	其他参数
1	卷积层	$3 \times 3 \times 32$	'SAME'
2	池化层	stride=2	Max Pooling
3	卷积层	$3 \times 3 \times 64$	'SAME'
4	池化层	stride=2	Max Pooling
5	卷积层	$3 \times 3 \times 64$	'SAME'
6	池化层	stride=2	Max Pooling
7	BatchNorm	-	-
8	Flatten	-	-
9	全连接层	$1024 \to 64$	-
10	全连接层	$64 \to 10$	-

表 1: *Cifar10* 数据集分类器网络结构

非线性函数均采用 ReLU 函数，最终的目标函数（损失函数）为

$$\arg\min \mathcal{L} = \text{CrossEntropyLoss}(\text{output}, \text{label})$$

20 轮迭代训练后，模型在验证集上的 Loss 及预测准确率变化如下：

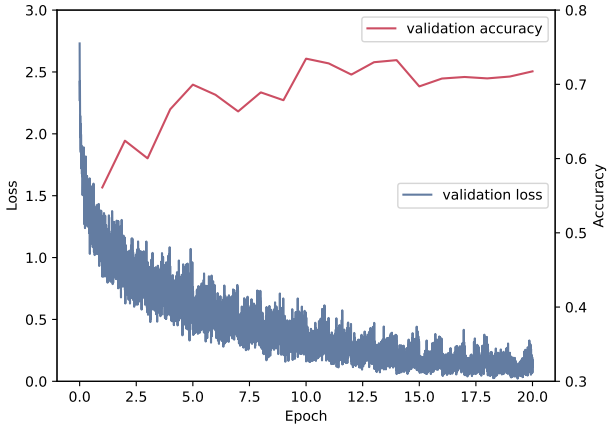


图 1: 在 *Cifar-10* 数据集上的训练结果

最终在验证集上的最高准确率为 **0.725900**。

实验中也发现使用'SAME' 的 padding 模式会比'VALID' 的 padding 模式取得更好的效果。这一点也符合直觉，由于图片本身尺寸就较小(32×32)，而 valid 模式会一定程度“舍弃”图片边缘像素的信息。

2. 在不平衡 Cifar-10 数据集上的训练

需要注意的是，在训练部分直接使用助教给出的掩码代码**是错误的做法**。由于 train_loader 会对原始数据集进行 shuffle 处理，因此只在训练部分使用掩码并不会导致模型训练中见到的样本数目发生变化，只是见到不同样本的概率有差异。在迭代轮数较大的情况下，实际上模型依然看到了原始数据集中的所有样本。

更加符合“不平衡数据集”模拟的做法应当是：在数据集上先划分 batch 并使用助教给出的掩码代码，将部分样本去除，得到真正的不平衡数据集。随后，使用去除后的数据集进行后续实验。这样即使 train_loader 对数据集进行了 shuffle 处理，数据集中各类别的样本数目依然是不平衡的。

在验证集上，模型的 20 轮迭代训练效果如下：（最终验证集上最高准确率为 **0.592400**。）

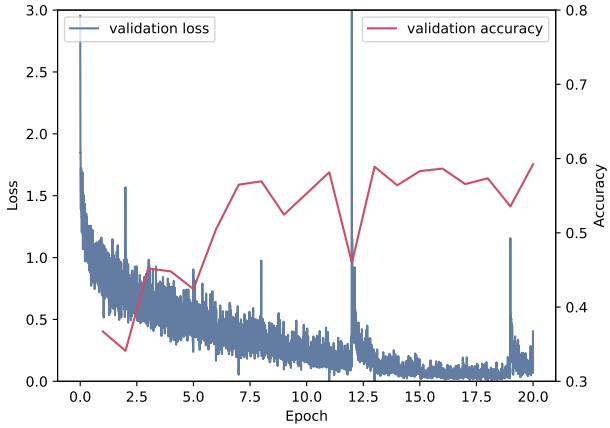


图 2: 在不平衡的 *Cifar-10* 数据集上的训练结果

3. 训练改进

笔者尝试了以下四种方式，试图改进模型在非平衡数据集上的训练效果。评估均在验证集上进行。

3.1. 简单复制样本副本以平衡各类别样本数

计算前五类（去除部分样本的类别）与后五类样本数均值的比值（约为 1:10）。将前五类的样本简单复制 9 份。以下简写为 copy。

3.2. 对复制的样本副本进行数据增强

我们进一步对复制的样本进行数据增强操作。具体而言，分别对前五类的样本复制后进行如下四种操作（包括原样本在内，共得到五份样本）：水平翻转、竖直翻转、改变亮度、改变对比度。

之后，再将前五类已有的样本直接复制一份，使十个类别的样本数尽可能相近。以下简写 aug。

3.3. 对损失函数进行加权修改

我们对 Cross Entropy Loss 进行加权修改：

$$loss_j = -\text{input}[j] + \log\left(\sum_{i=0}^{K} \exp\left(\text{input}[i]\right)\right)$$

$$\mathcal{L} = \sum_{j=1}^{K} w_j \cdot loss_j, \quad w_j = \frac{\sum_{i=1}^{K} N_i}{N_j}$$

其中 w_j 为加权系数，N_j 为第 j 类的样本数。该种方式以下简写为 loss。

3.4. 同时结合数据增强和加权损失函数

在 **3.2** 中，我们仍然进行了简单的复制操作。为改进这一点，我们可以在对前五类数据进行数据增强得到五份不同的样本后，直接在该数据集上使用加权的损失函数。该方式简写为 aug+loss。

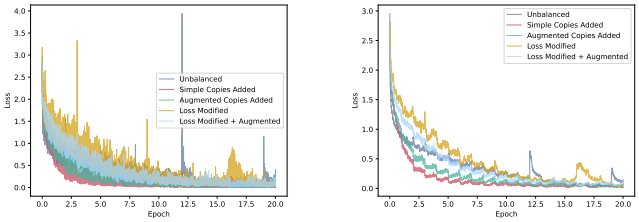
3.5. 改进效果评估

使用以上四种方式进行改进后，验证集上的最高准确率对比如表2。模型训练中 Loss 和准确率变化如图3和4。(均为矢量图，可放大查看细节。)

方式	最高准确率
无处理	0.592400
copy	0.582000
aug	0.605500
loss	**0.628000**
loss+aug	0.606000

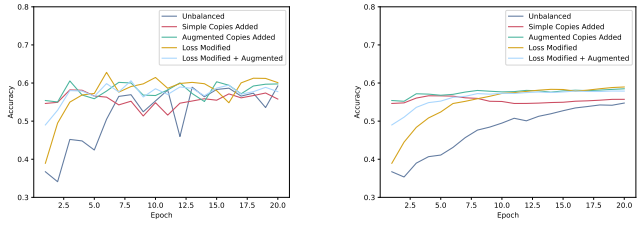
表 2: 验证集上最高准确率对比

为了更好地看出各种方式下训练效果的变化趋势，笔者在作图时对原始数据进行数据平滑处理后再可视化，得到图3(b) 和4(b)。



(a) Loss (b) Smoothed Loss

图 3: 验证集上 *Loss* 变化（平滑数据处理前后）



(a) Acc (b) Smoothed Acc

图 4: 验证集上准确率变化（平滑数据处理前后）

由以上结果可见，四种方式都能较好地提升模型的效果，其中**加权损失函数效果最好**。整体而言，四种方式对效果的提升表现为：使用加权损失函数 > 使用加权损失函数 + 对复制副本进行数据增强 > 对复制副本进行数据增强 > 简单复制样本（该方法与增加训练轮数效果类似）。

由图4(b) 可见，直接简单复制样本和对复制样本进行数据增强两种改进方式下，模型收敛较快；修改损失函数后模型收敛速度相对较慢，但最终准确率的提升较高。

但无论使用何种方式进行模型训练的改进，最终都无法复现平衡数据集上的效果。这一现象一定程度上是在模拟不平衡数据集时删除了较多的训练样本导致的。

Deep Learning Homework 03：拼图问题与分类模型预训练

520030910155 邱一航

1. 在 Cifar-10 数据集上进行拼图任务

首先对原始数据集进行处理。笔者使用的图像划分方式为 2×2，对每张图的四个子图进行重新排序后生成新的数据集。

一个划分示例如下：

(a) Before rearrangment (b) After rearrangment

图 1: 子图划分示例

考虑到后续该任务使用的网络模型会被用于作为 Cifar10 数据集上分类器的预训练，因此网络结构应与 task 2 中使用的网络结构尽可能相同。

因此，笔者搭建的网络结构如表1。（对卷积层，其他参数指 padding 模式；对池化层，其他参数指池化层性质（Mean/Max Pooling））

id	性质	核尺寸 × 通道数	其他参数
1	卷积层	$3 \times 3 \times 32$	'SAME'
2	池化层	stride=2	Max Pooling
3	卷积层	$3 \times 3 \times 64$	'SAME'
4	池化层	stride=2	Max Pooling
5	卷积层	$3 \times 3 \times 64$	'SAME'
6	BatchNorm		-
7	Flatten		-
8	全连接层	$1024 \to 64$	-
9	全连接层	$64 \to 256$	-
将四张子图的输出拼接为 1024 维向量			
10	全连接层	$1024 \to 256$	-
11	全连接层	$256 \to 16$	-

表 1: *Cifar10* 数据集上拼图模型网络结构

其中，横线上方的网络结构与 task 2 中的网络结构几乎一致（只去除了一个池化层并改变了最后一层全连接层的参数量）。最终，将输出的 16 维向量按顺序重新变形为 4×4 矩阵后通过 Sinkhorn 算法变为双随机矩阵，将双随机矩阵与真值（排列阵）取 Binary Cross Entropy Loss 作为整个模型的损失函数。

记网络的输出变形后经 Sinkhorn 算法得到的双随机矩阵为 Y，排列阵真值为 P。简单起见，模型预测的子图排列顺序通过对 Y 取行最大值序号的方式确定。（这样生成的排列顺序中可能出现重复的序号，所以使用更好的方式确定排列顺序能够进一步提升模型的拼图效果。）

网络结构总览图如下。

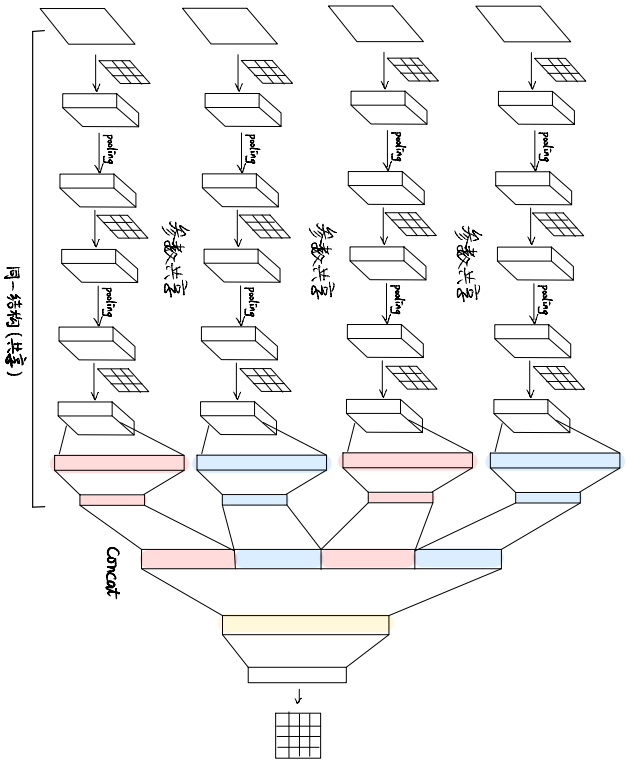


图 2: *Cifar10* 拼图任务模型示意图

笔者使用以下三个指标来评估模型拼图效果。

- 准确率。将模型预测的子图排列顺序与真值比对，某一位置的预测值与真值相同则认为该位置正确。统计每个位置正确的概率。
- 完全准确率。将模型预测的子图排列顺序与真值比对，只有所有位置都正确时才认为当前样本预测正确，统计子图排列顺序完全正确的概率。
- 置信度。Confidence $= Y \bigodot P$. 该指标的物理含义是模型预测结果中每个子图出现在正确位置的概率。

10 轮迭代训练后，模型在验证集上的 Loss 及预测准确率、完全准确率、置信度变化如图3。

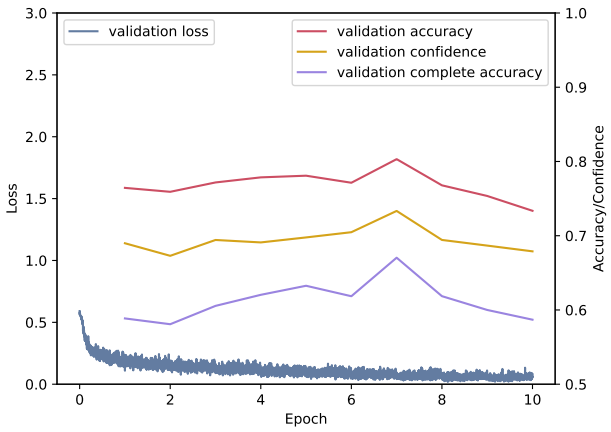


图 3: *Cifar-10* 数据集拼图模型训练结果

最终在验证集上的最高准确率为 **0.791800**，最高完全准确率为**0.659300**，最高置信度为**0.730900**。

2. 作为预训练移植至 task 2

由于笔者设计的拼图模型的网络结构与 task 2 中 Cifar10 数据集分类器的网络结构有相同的部分，因此可以将拼图模型的训练作为预训练，直接将训练后的拼图模型的部分参数移植到 task 2 的分类器网络中作为初始化。

具体迁移方式如表2，pretrain 为 Y 的网络层为载入拼图模型训练后参数的网络层。

id	性质	核尺寸 × 通道	其他参数	pretrain
1	卷积层	$3 \times 3 \times 32$	'SAME'	Y
2	池化层	stride=2	Max	-
3	卷积层	$3 \times 3 \times 64$	'SAME'	Y
4	池化层	stride=2	Max	-
5	卷积层	$3 \times 3 \times 64$	'SAME'	Y
6	池化层	stride=2	Max	-
7	BatchNorm		-	-
8	Flatten		-	-
9	全连接层	$1024 \rightarrow 64$	-	Y
10	全连接层	$64 \rightarrow 10$	-	N

表 2: *Cifar10* 数据集分类器网络结构

迁移前后，Cifar10 数据集上的分类器训练效果对比如下图（图4），其中准确率为验证集上的准确率。使用拼图模型作为预训练前后，验证集上最高准确率分别为**0.725900** 和**0.742900**。

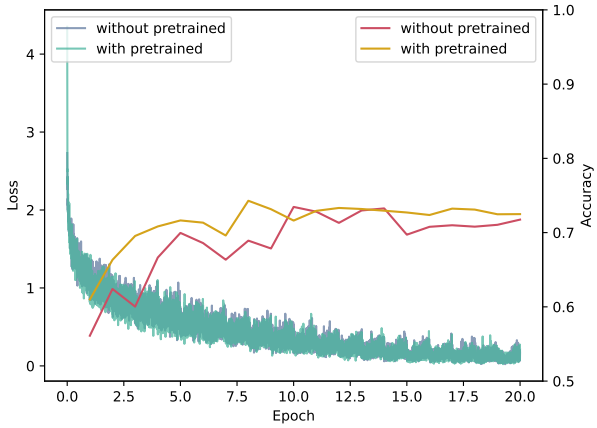


图 4: *Cifar-10* 数据集拼图模型训练结果

可以发现，使用拼图任务作为预训练可以一定程度地提高分类器在 Cifar10 数据集上的效果。