比较 Cutout，Mixup，Cutmix 在训练卷积神经网络上的性 1

顾淳 李优泉

概述

卷积神经网络可以很好的抓取图像中的表征信息，但是由于模型的容量大，训练很容易趋于过拟合。在本次实验中，我们比较 5 种数据增强方法：baseline，baseline+，cutout，mixup，cutmix 在训练复杂卷积神经网络的性能，得到的结果为：在 Resnet18、Densenet、VGG 上，cutmix 有自豪的性能；在 Lenet 上，简单数据增强[1]有最好的性能。并且对于 cutout 和 mixup 来说，当分类类数较少时，cutout 有较好性能，当分类类数较多时，mixup 有较好性能。在 CIFAR10 上，我们得到的最好的模型是于 Resnet18 上用 cutmix 进行数据增强，准确率达到 96.20%；在 CIFAR100 上，我们得到的最好的模型同样是于 Resnet18 上用 cutmix 进行数据增强，准确率达到 79.77%。

本次实验的代码地址：

https://github.com/SuLvXiangXin/cutout-mixup-cutmix

1.相关工作 介绍网络

1. Densenet

研究表明，如果靠近输入和靠近输出的层之间包含更短的连接，那么卷积网络可以在很大程度上更深入、更准确和高效地进行训练。密集卷积网络(DenseNet)，它以前馈的方式将每一层与每一层连接起来。传统的有 L 层的卷积网络有 L 个连接，每个层和它的后续层之间有一个连接，而我们的网络有 L(L+1)/2 个直接连接。对每一层，所有前一层的特征图都被用作输入，而它自己的特征图被用作所有后续层的输入。Densenet 有几个令人信服的优势：它们缓解了消失梯度问题，加强了特性传播，鼓励特性重用，并大大减少了参数的数量。

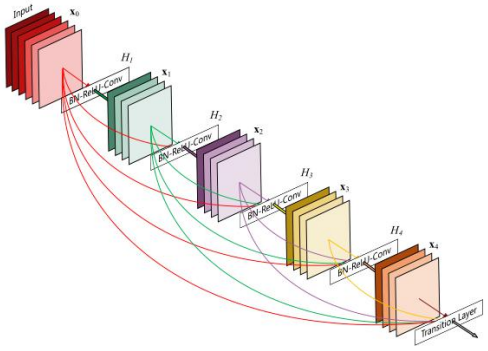

图 1：Densenet

2. Lenet

1998 年，LeCun 等人发布了 LeNet 网络，从而揭开了深度学习的面纱，之后的深度神经网络都是在这个基础之上进行改进的，其结构如图所示。

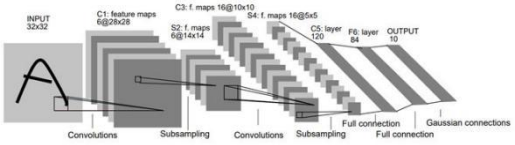

图 2：Lenet

如图 2，LeNet-5（因为有 5 个卷积层而得名）是由卷积层、池化层、全连接层的顺序连接，网络中的每个层使用一个可微分的函数将激活数据从一层传递到另一层。LeNet-5 开创性的利用卷积从直接图像中学习特征，在计算性能受限的当时能够节省很多计算量，同时也指出卷积层的使用可以保证图像的空间相关性（也是基于此，之后的一些网络开始慢慢摒弃全连接层，同时全连接层还会带来过拟合的风险）。

3. Resnet

ResNet（Residual Neural Network）由微软研究院的 Kaiming He 等四名华人提出，通过使用 ResNet Unit 成功训练出了 152 层的神经网络，并在 ILSVRC2015 比赛中取得冠军，在 top5 上的错误率

1 在每条边先填充 4 个 0 像素，然后再裁剪为 32x32，在按 50%概率水平翻转。

为 3.57%，同时参数量比 VGGNet 低，效果非常突出。ResNet 的结构可以极快的加速神经网络的训练，模型的准确率也有比较大的提升。同时 ResNet 的推广性非常好，甚至可以直接用到 InceptionNet 网络中。ResNet 的主要思想是在网络中增加了直连通道，即 Highway Network 的思想。此前的网络结构是性能输入做一个非线性变换，而 Highway Network 则允许保留之前网络层的一定比例的输出。ResNet 的思想和 Highway Network 的思想也非常类似，允许原始输入信息直接传到后面的层中，如图 3 所示。在本次实验中，我们使用的是带有 17 个卷积层和 1 个全连接层的 Resnet18。

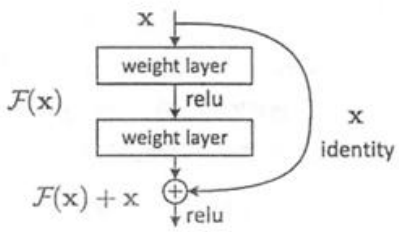

图 3：Resnet

4. Vgg

VGGNet 由牛津大学计算机视觉组合和 Google DeepMind 公司研究员一起研发的深度卷积神经网络。它探索了卷积神经网络的深度和其性能之间的关系，通过反复的堆叠 3*3 的小型卷积核和 2*2 的最大池化层，成功的构建了 16~19 层深的卷积神经网络。VGGNet 获得了 ILSVRC 2014 年比赛的亚军和定位项目的冠军，在 top5 上的错误率为 7.5%。目前为止，VGGNet 依然被用来提取图像的特征。

VGGNet 全部使用 3*3 的卷积核和 2*2 的池化核，通过不断加深网络结构来提升性能。网络层数的增长并不会带来参数量上的爆炸，因为参数量主要集中在最后三个全连接层中。同时，两个 3*3 卷积层的串联相当于 1 个 5*5 的卷积层，3 个 3*3 的卷积层串联相当于 1 个 7*7 的卷积层，即 3 个 3*3 卷积层的感受野大小相当于 1 个 7*7 的卷积层。但是 3 个 3*3 的卷积层参数量只有 7*7 的一半左右，同时前者可以有 3 个非线性操作，而后者只有 1 个非线性操作，这样使得前者对于特征的学习能力更强。

其结构如图 4 所示：

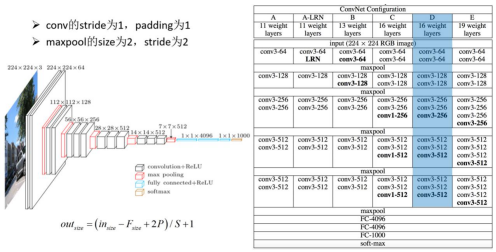

图 4：VGG

2.介绍方法

1.baseline(+)

介绍

在本次实验中，我们构造了两个 baseline，一个是没有使用数据增强的 baseline，一个是使用简单数据增强处理后的 baseline+。

原理

baseline+中的简单数据增强就是先在图片的每条边上填充 4 个像素的 padding，再在得到的 40x40 的图片中随机截取 32x32 的区域，最后再按 50%的概率随机水平翻转，得到的结果则作为训练集图片。

```
train_transform.transforms.append(transforms.RandomCrop(32, padding=4))
train_transform.transforms.append(transforms.RandomHorizontalFlip())
```

图 5 ：baseline+ 简单数据增强关键代码

图 6：baseline+ 简单数据增强可视化

2.Mixup

介绍

mixup 是一种运用在计算机视觉中的对图像进行混类增强的算法，它可以将不同类之间的图像进行混合，分类结果按比例分配，从而扩充训练数据集。

原理

假设$batch_{x_1}$是一个 batch 样本，$batch_{y_1}$是该 batch 样本对应的标签；$batch_{x_2}$是另一个 batch 样本，

$batch_{y_2}$是该 batch 样本对应的标签，λ是由参数为α，β的贝塔分布计算出来的混合系数，由此我们可以得到 mixup 原理公式为：

$$\lambda = \mathrm{Beta}(\alpha, \beta)$$

$$\mathrm{mixup\_batch_x} = \lambda * \mathrm{batch_{x1}} + (1-\lambda) * \mathrm{batch_{x2}}$$

$$\mathrm{mixup\_batch_y} = \lambda * \mathrm{batch_{y1}} + (1-\lambda) * \mathrm{batch_{y2}}$$

其中 Beta 指的是贝塔分布，$\mathrm{mixup_batch_x}$是混合后的 batch 样本，$\mathrm{mixup_batch_y}$是混合后的 batch 样本对应的标签

```
inputs, targets_a, targets_b, lam = mixup_data(inputs, targets, args.alpha, args.cuda)
inputs, targets_a, targets_b = map(Variable, (inputs, targets_a, targets_b))
model.zero_grad()
outputs = model(inputs)
loss = mixup_criterion(criterion, outputs, targets_a, targets_b, lam)
```

图 7：mixup 关键代码

```
def mixup_data(x, y, alpha=1.0, use_cuda=True):
    '''Returns mixed inputs, pairs of targets, and lambda'''
    if alpha > 0:
        lam = np.random.beta(alpha, alpha)
    else:
        lam = 1

    batch_size = x.size()[0]
    if use_cuda:
        index = torch.randperm(batch_size).cuda()
    else:
        index = torch.randperm(batch_size)

    mixed_x = lam * x + (1 - lam) * x[index, :]
    y_a, y_b = y, y[index]
    return mixed_x, y_a, y_b, lam


def mixup_criterion(criterion, pred, y_a, y_b, lam):
    return lam * criterion(pred, y_a) + (1 - lam) * criterion(pred, y_b)
```

图 8：mixup 关键代码

图 9：cutout 可视化

4.Cutout

介绍

Cutout 是一种简单的卷积神经网络正则化方法，它在训练过程中随机地将样本中的部分区域 cut 掉，并且在该区域填充 0，分类的结果不变，可以用来提高卷积神经网络的鲁棒性和整体性能。

原理

在训练过程中随机地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

```
    Args:
        img (Tensor): Tensor image of size (C, H, W).
    Returns:
        Tensor: Image with n_holes of dimension length x length cut out of it.
    """
    h = img.size(1)
    w = img.size(2)
    mask = np.ones((h, w), np.float32)
    for n in range(self.n_holes):
        y = np.random.randint(h)
        x = np.random.randint(w)
        y1 = np.clip(y - self.length // 2, 0, h)
        y2 = np.clip(y + self.length // 2, 0, h)
        x1 = np.clip(x - self.length // 2, 0, w)
        x2 = np.clip(x + self.length // 2, 0, w)
        mask[y1: y2, x1: x2] = 0.
    mask = torch.from_numpy(mask)
    mask = mask.expand_as(img)
    img = img * mask
```

图 10：cutout 关键代码

图 11：cutout 可视化

5.Cutmix

介绍

Cutmix 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

原理

x_A和x_B是两个不同的训练样本，y_A和y_B是对应的标签值，CutMix 需要生成的是新的训练样本和对应标签：x'和 y'，公式如下：

$$x' = M \odot x_A + (1-M) \odot x_B$$

$$y' = \lambda y_A + (1-\lambda) y_B$$

$M \in \{0,1\}^{W*H}$是为了 drop 掉部分区域和进行填充的二进制掩码，⊙是逐像素相乘，1 是所有元素都为 1 的二进制掩码，λ和 Mixup 一样属于 Beta 分布：$\lambda \sim \mathrm{Beta}(\alpha, \alpha)$，令$\alpha$=1 则$\lambda$服从（0，1）均匀分布

为了对二进制掩码 M 进行采样，首先要对剪裁区域的边界框 $B = (r_x, r_y, r_w, r_h)$进行采样，用来对样本x_A和x_B做剪裁区域的指示标定。在论文中对矩形掩码 M 进行采样（长宽与样本大小成比例）。

剪裁区域的边界框采样公式如下：

$$r_x \sim \mathrm{Unif}(0, W),\ r_w = W\sqrt{1-\lambda}$$

$$r_y \sim \mathrm{Unif}(0, H),\ r_h = H\sqrt{1-\lambda}$$

保证剪裁区域的比例为$\frac{r_w r_h}{WH} = 1 - \lambda$，确定好裁剪区域 B 之后，将二进制掩码 M 中的裁剪区域 B 置 0，其他区域置 1。就完成了掩码的采样，然后将样本 A 中的裁剪区域 B 移除，将样本 B 的裁剪区域 B 进行裁剪然后填充到样本 A。

```
lam = np.random.beta(args.alpha, args.alpha)
rand_index = torch.randperm(inputs.size()[0]).cuda()
target_a = targets
target_b = targets[rand_index]
bbx1, bby1, bbx2, bby2 = rand_bbox(inputs.size(), lam)
inputs[:, :, bbx1:bbx2, bby1:bby2] = inputs[rand_index, :, bbx1:bbx2, bby1:bby2]
# adjust lambda to exactly match pixel ratio
lam = 1 - ((bbx2 - bbx1) * (bby2 - bby1) / (inputs.size()[-1] * inputs.size()[-2]))
# compute output
output = model(inputs)
loss = mixup_criterion(criterion, output, target_a, target_b, lam)
```

图 12：cutmix 关键代码

```
def rand_bbox(size, lam):
    W = size[2]
    H = size[3]
    cut_rat = np.sqrt(1. - lam)
    cut_w = np.int(W * cut_rat)
    cut_h = np.int(H * cut_rat)

    # uniform
    cx = np.random.randint(W)
    cy = np.random.randint(H)

    bbx1 = np.clip(cx - cut_w // 2, 0, W)
    bby1 = np.clip(cy - cut_h // 2, 0, H)
    bbx2 = np.clip(cx + cut_w // 2, 0, W)
    bby2 = np.clip(cy + cut_h // 2, 0, H)

    return bbx1, bby1, bbx2, bby2


def mixup_criterion(criterion, pred, y_a, y_b, lam):
    return lam * criterion(pred, y_a) + (1 - lam) * criterion(pred, y_b)
```

图 13：cutmix 关键代码

图 14：cutmix 可视化

3.实验

0.介绍数据集

CIFAR-10

CIFAR-10 数据集由 10 个类的 60000 个 32x32 彩色图像组成，每个类有 6000 个图像。有 50000 个训练图像和 10000 个测试图像。

数据集分为五个训练批次和一个测试批次，每个批次有 10000 个图像。测试批次包含来自每个类别的恰好 1000 个随机选择的图像。训练批次以随机顺序包含剩余图像，但一些训练批次可能包含来自一个类别的图像比另一个更多。总体来说，五个训练集之和包含来自每个类的正好 5000 张图像。

以下是数据集中的类，以及来自每个类的 10 个随机图像：

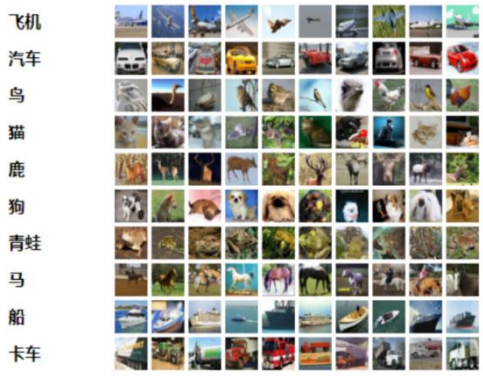

图 15：CIFAR10 数据集

这些类完全相互排斥。汽车和卡车之间没有重叠。“汽车”包括轿车，SUV，这类东西。“卡车”只包括大卡车。都不包括皮卡车。

CIFAR-100

这个数据集就像 CIFAR-10，除了它有 100 个类，每个类包含 600 个图像。，每类各有 500 个训练图像和 100 个测试图像。CIFAR-100 中的 100 个类被分成 20 个超类。每个图像都带有一个“精细”标签（它所属的类）和一个“粗糙”标签（它所属的超类）

图 16 是 CIFAR-100 中的类别列表：

超类	类别
水生哺乳动物	海狸，海豚，水獭，海豹，鲸鱼
鱼	水族馆的鱼，比目鱼，射线，鲨鱼，鳟鱼
花卉	兰花，罂粟花，玫瑰，向日葵，郁金香
食品容器	瓶子，碗，罐子，杯子，盘子
水果和蔬菜	苹果，蘑菇，橘子，梨，甜椒
家用电器	时钟，电脑键盘，台灯，电话机，电视机
家用家具	床，椅子，沙发，桌子，衣柜
昆虫	蜜蜂，甲虫，蝴蝶，毛虫，蟑螂
大型食肉动物	熊，豹，狮子，老虎，狼
大型人造户外用品	桥，城堡，房子，路，摩天大楼
大自然的户外场景	云，森林，山，平原，海
大杂食动物和食草动物	骆驼，牛，黑猩猩，大象，袋鼠
中型哺乳动物	狐狸，豪猪，负鼠，浣熊，臭鼬
非昆虫无脊椎动物	螃蟹，龙虾，蜗牛，蜘蛛，蠕虫
人	宝贝，男孩，女孩，男人，女人
爬行动物	鳄鱼，恐龙，蜥蜴，蛇，乌龟
小型哺乳动物	仓鼠，老鼠，兔子，母老虎，松鼠
树木	枫树，橡树，棕榈，松树，柳树
车辆1	自行车，公共汽车，摩托车，皮卡车，火车
车辆2	割草机，火箭，有轨电车，坦克，拖拉机

图 16：CIFAR100 数据集

dataset	model	base	base+	cutout	mixup	cutmix
cifar10	resnet18	88.97±0.73	94.99±0.24	95.90±0.28	95.77±0.46	96.20±0.48
	lenet	63.63±3.15	75.15±1.45	71.60±1.28	74.95±0.48	70.01±0.47
	densenet	91.23±0.28	94.48±0.22	95.21±0.30	95.07±0.19	95.79±0.23
	vgg	87.64±0.30	92.26±0.51	92.87±0.31	92.43±0.45	93.26±0.43
cifar100	resnet18	63.26±0.78	77.50±0.88	77.63±0.81	79.03±0.41	79.77±0.58
	lenet	24.33±1.30	41.32±0.81	37.30±1.18	40.68±0.95	35.55±0.19
	densenet	70.13±0.53	75.94±0.98	76.88±0.51	77.23±0.58	77.79±0.38
	vgg	61.30±0.71	71.12±0.66	69.93±0.16	71.99±0.38	73.78±0.86

表 1：5 种数据增强方法，4 种网络结构，2 个数据集上的测试准确率(%)。每个结果来自 5 次重复的训练，并且给出了 95%的置信区间。

1. 训练

实验环境为 Google 的 Colab，GPU 使用的是 Colab 中的 Tesla P100。

我们共使用五种方法来训练神经网络，分别是 baseline，baseline+，mixup，cutout，cutmix。

在 mixup，cutout，cutmix 三种方法中，我们均以 baseline+中的简单数据增强方法为基础，再在其上进行新的数据增强。

为了测试这几种方法，我们在四种网络结构上训练模型，分别是：Resnet18、Lenet、Densenet、VGG。

对于所有的训练，我们使用了相同训练过程。我们设置 batch_size=128，使用交叉熵损失函数和随机梯度下降方法（SGD）训练 200 个 Epoch。我们把 learning_rate 设置为 0.1，并设置 scheduler 使得其在第 60、120、160 个 Epoch 处下降为原来的 0.1 倍。为了找到每组实验最好的超参数，我们把训练集中的 20%作为验证集。

值得一提的是，为了确定 cutmix 中每个 batch 进行 cutmix 操作的概率，我们通过测试 cutmix_prob 取不同值时在网络上的性能，最终确定 cutmix_prob 取 100%为最优。如图 17。

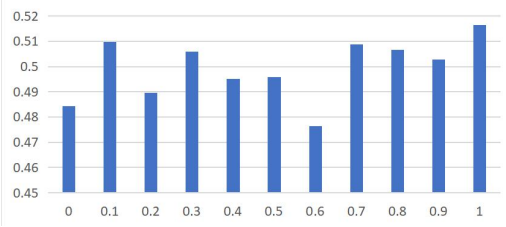

图 17：以 Reset18 为网络结构，cutmix_prob 从 0 到 1 取 11 个值分别训练 10 个 Epoch。横坐标为 cutmix_prob,纵坐标为准确率。

2. 分析

实验结果如表 1。根据表 1 中的实验结果我们可以看到在 Resnet18、Densenet、VGG 上，cutmix 方法拥有最好的性能；而在 lenet 上，简单数据增强方法反而相对于其他三种复杂的数据增强方法有更好的性能，这可能是源于 Lenet 网络容量较小，用复杂的数据进行训练会导致模型欠拟合，损失函数下降困难。

在 CIFAR10：Resnet18、Densenet、VGG 上，cutout 在简单数据增强的基础上提升了 0.91%，0.73%，0.61%，mixup 在简单数据增强的基础上提升了 0.78%，0.59%，0.17%，cutmix 在简单数据增强的基础上提升了 1.21%，1.31%，1.00%。

在 CIFAR10：Resnet18、Densenet、VGG 上，cutout 在简单数据增强的基础上提升了 0.13%，0.94%，-1.19%，mixup 在简单数据增强的基础上提升了 1.53%，1.29%，0.87%，cutmix 在简单数据增强的基础上提升了 2.27%，1.85%，2.66%。

4.结论

综合以上的实验结果，cutmix 是当之无愧的 3 种数据增强方法中最好的，它同时体现了 cutout 和 mixup 的优点，在复杂神经网络上有效的防止了过拟合的发生，在 CIFAR10 和 CIFAR100 中都体现出相对于前几种方法较大的进步。遗憾的是，我们并没有很好地体现出 Lenet 的性能，可能的原因是没有找到一组合适的超参数在 CIFAR 数据集上训练。从结果来看，几种方法在 CIFAR10 数据集上已经表现的很好了，但是在 CIFAR100 上，我们还可以有更好的网络和数据增强方法对准确率实现更进一步的

提升。

Reference

[1] Hongyi Zhang, Moustapha Cisse, Yann N Dauphin, and David Lopez-Paz. mixup: Beyond empirical risk minimization.
arXiv preprint arXiv:1710.09412, 2017.

[2] Terrance DeVries and Graham W Taylor. Improved regularization of convolutional neural networks with cutout.
arXiv preprint arXiv:1708.04552, 2017.

[3] Sangdoo Yun, Dongyoon Han, Sanghyuk Chun, Youngjoon Yoo and Clova AI Research, NAVER Corp. CutMix: Regularization Strategy to Train Strong Classifiers with Localizable Features.
arXiv preprint arXiv: 1905.04899,2019.

[4] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks.
In NIPS, 2012.

[5] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. Journal of Machine Learning Research, 15:1929–1958, 2014.

Appendix

五种方法在两个数据集、四种网络结构上的测试曲线。