论文复现总结

1 论文概述

这篇论文提出了一种名为Remix的新的正则化技术，旨在解决数据不平衡对深度图像分类器性能的影响。在传统的Mixup技术的基础上，Remix放宽了Mixup的约束，并使特征和标签的混合因子能够解耦。具体而言，在混合两个样本时，Remix将特征以与Mixup相同的方式进行混合，但通过为少数类别分配不成比例的较高权重来偏向少数类别的标签。通过这样的方式，分类器学习将决策边界向多数类别推进，平衡多数类别和少数类别之间的泛化误差。

论文对Mixup、Manifold Mixup和CutMix等现有的正则化技术在数据不平衡情况下进行了研究，并表明所提出的Remix在构建的CIFAR-10、CIFAR-100、CINIC-10等不平衡数据集上明显优于这些现有技术以及几种重新加权和重新采样技术。同时，论文还在真实世界的大规模不平衡数据集iNaturalist 2018上评估了Remix。实验结果证实，Remix相对于先前的方法提供了一致且显著的改进。

2 核心算法

Remix算法通过将特征和标签混合来处理数据。具体而言，Remix使用以下方法将两个样本的特征和标签进行混合：

(1) 特征混合：

$$MixedFeature = \lambda_x \cdot Feature_1 + (1 - \lambda_x) \cdot Feature_2 \tag{1}$$

其中，λ_x 是特征混合因子，取值范围为0到1，表示两个样本特征的混合程度。

(2) 标签混合：

$$MixedLabel = \lambda_y \cdot Label_1 + (1 - \lambda_y) \cdot Label_2 \tag{2}$$

其中，λ_y 是标签混合因子，取值范围为0到1，表示两个样本标签的混合程度，用于调整混合标签时对少数类别的偏好程度。

λ_x 从beta分布中抽样得到，该方法定义λ_y的确切形式如式(3)所示

$$\lambda_y = \begin{cases} 0, & n_i/n_j \geq \kappa \text{ and } \lambda_x < \tau \\ 1, & n_i/n_j \leq 1/\kappa \text{ and } 1-\lambda_x < \tau \\ \lambda_x, & \text{otherwise} \end{cases} \tag{3}$$

其中n_i和n_j表示样本i和样本j对应的分类的样本数量。κ和τ是该方法中的两个超参数。

3 复现结果

分别在 Imbalanced cifar-10 和 Imbalanced cifar-100 上进行了实验，实验结果与源论文结果的对比如表 1和表 2所示。可以发现，对于纯Remix方法，复现结果与源论文结果大致相同，无明显差异。对于Remix混合方法，由于原文未细说重采样与重加权的详细方法，我的复现结果与原文有一些差异。

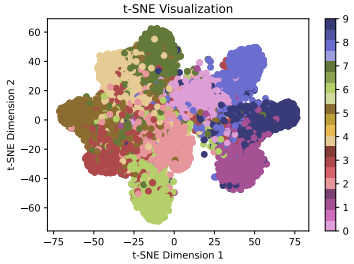
Table 1: Results on Imbalanced cifar-10

Dataset	**Imbalanced CIFAR-10**							
Imbalance Type	long-tailed				step			
Imbalance Ratio	100		10		100		10	
Reproduct/Source	R	S	R	S	R	S	R	S
Remix	**75.05**	75.36	**89.16**	88.15	**69.04**	68.98	**87.74**	86.34
Remix-RS	**74.72**	76.23	**88.02**	87.7	**68.84**	67.28	**87.39**	86.63
Remix-RW	**74.56**	75.1	**89.53**	87.91	**68.98**	68.74	**87.93**	86.38

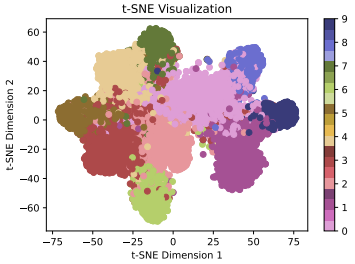
Table 2: Results on Imbalanced cifar-100

Dataset	**Imbalanced CIFAR-100**							
Imbalance Type	long-tailed				step			
Imbalance Ratio	100		10		100		10	
Reproduct/Source	R	S	R	S	R	S	R	S
Remix	**42.15**	41.94	**59.48**	59.36	**39.7**	39.96	**56.97**	57.06
Remix-RS	**41.18**	41.13	**59.58**	58.62	**39.42**	39.74	**57.07**	56.09
Remix-RW	**41.11**	33.51	**59.46**	57.65	**39.22**	17.42	**56.45**	54.45

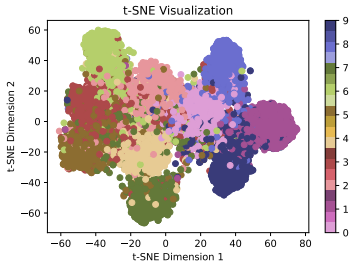
用tsne分别将Mixup与ReMix模型提取的特征降维至2维，并绘制了各类样本的分布情况，如图 1所示。纵向对比，由于样本数较多的类占优势，可以看到，模型会将较多'9'、'8'类的样本误判成'0'、'1'类。横向对比，不能很明显地看出Mixup与ReMix的差别，但从准确率上看，ReMix相对Mixup有较明显的提高。



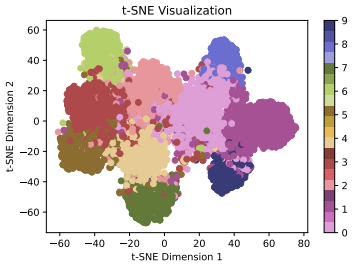
(a) True value distribution using Mixup



(b) Prediction distribution using Mixup



(c) True value distribution using ReMix



(d) Prediction distribution using ReMix

Figure 1: Sample distribution of each class

4 附录

4.1 核心代码

```
#Remix混合数据
def remix_data(x, y, alpha, args):
    #beta分布，随机生成lam_x
        if alpha > 0:
            lam = np.random.beta(alpha, alpha)
        else:
            lam = 1

        batch_size = x.size()[0]
    #随机配对,混合特征
        if args.gpu != None:
            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]
        lam_y = torch.zeros_like(y, dtype = torch.float64)
    #对于每个混合样本，根据公式确定lam_y
        for i in range(batch_size):
            if args.cls_num_list[y_a[i]] / args.cls_num_list[y_b[i]] >= \
            args.k and lam < args.t:
                lam_y[i] = 0.
            elif args.cls_num_list[y_a[i]] / args.cls_num_list[y_b[i]] <= 1 \
            / args.k and lam > 1 - args.t:
                lam_y[i] = 1.
            else:
                lam_y[i] = lam
        return mixed_x, y_a, y_b, lam ,lam_y
#混合损失
def remix_criterion(criterion, pred, y_a, y_b, lam, lam_y):
        res = 0
        batch_size = y_a.size()[0]
        for i in range(batch_size):
            res += lam_y[i] * criterion(pred[[i]], y_a[[i]]) \
            + (1 - lam_y[i]) * criterion(pred[[i]], y_b[[i]])
        return res / batch_size
```

4.2 完整代码

https://github.com/1372172287/Remix/tree/master