Homework 3：图像复原任务

522030910135 陈元杰

1. 数据集构建

我们基于 CIFAR10 数据集，对于数据集中的每一张图片，将它切分成若干子图，同时将这些子图随机打乱顺序并重新拼接成新的图片。将打乱后的每张子图对应的在原图中的顺序作为标签。我们的任务是对于打乱后的图片，找到正确的顺序将它们拼接成原图，如下图所示：

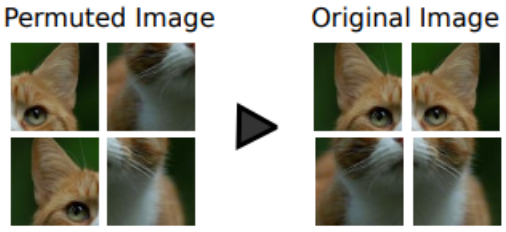

图 1: 图像复原任务

若将原图分成形状为 $m \times n$ 的子图，我们定义第 i 行，第 j 列（都从 0 开始计数）位置的子图的编号为

$$p(i,j) = i \times n + j$$

按照这样的方式，我们可以将子图编号为 $0, 1, \ldots, m \times n - 1$。对于这些子图进行一次随机 shuffle，作为打乱之后的标签，然后按照 shuffle 之后的编号顺序将这些子图重新拼接成新的图片，将这些图片和标签作为数据集。以将图像切分成 2×2 的子图为例。对于 CiFAR10 彩色数据集来说，每张子图大小的规模为 $(3, 16, 16)$。训练集中共有 50000 张图片，我们将所有训练集的图片拼接为矩阵，形状为 $(50000, 4, 3, 16, 16)$。测试集中共有 10000 张图片，因此测试集的图片矩阵形状为 $(10000, 4, 3, 16, 16)$。

2. 模型结构

2.1. 神经网络结构

我们的主要想法是对于每张子图利用卷积神经网络提取它的特征，然后利用子图的特征预测可能的正确位置，这类似于图像分类，每一个可能的位置可以看成一个类别。参考论文中 DeepPermNet 的网络结构，如下图所示：

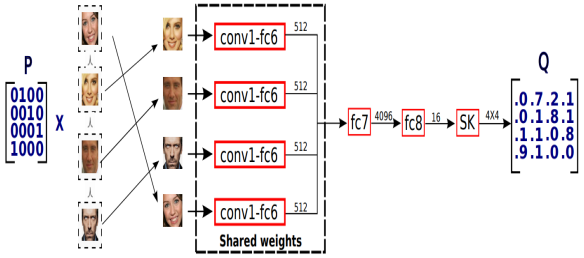

图 2: *DeepPermNet*

参考上图中的网络结构，我们先设计一个卷积神经网络用于提取每张子图的特征。我们设计了三层不同大小卷积核的卷积层，并以 RELU 作为激活函数，并进行最大池化。同时在经过每个卷积层后，我们还会进行批归一化处理，加速收敛速度，减小过拟合。我们的网络结构代码如下所示：

```
        self.layer1 = nn.Sequential(
                nn.Conv2d(3,32,(5,5),padding=2),
                nn.BatchNorm2d(32),
                nn.ReLU(),
                nn.MaxPool2d(2),
                nn.Conv2d(32,16,(3,3),padding=1),
                nn.BatchNorm2d(16),
                nn.ReLU(),
                nn.MaxPool2d(2),
                nn.Conv2d(16,64,(5,5),padding=2),
                nn.BatchNorm2d(64),
                nn.ReLU(),
                nn.Flatten(),
                nn.Linear(64*patch_size[0]*patch_size[1]//16, 512),
                nn.ReLU()
        )
```

上面的卷积神经网络可以将提取每张子图中的特征，然后用这些特征经过线性层，生成对于不同正确位置的预测概率。

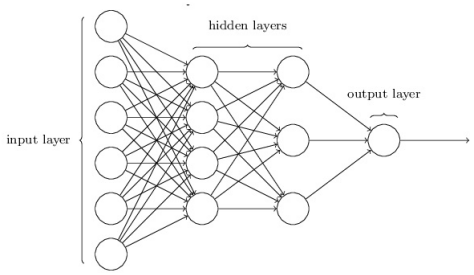

图 3: 线性层

这一层的网络结构代码如下所示：

```
        self.layer2 = nn.Sequential(
            nn.Flatten(),
            nn.Linear(self.patch_num*512, 4096),
            nn.ReLU(),
            nn.Linear(4096, self.patch_num**2)
        )
```

为了将经过网络之后的结果转变为预测概率，我们利用 Sinkhorn 算法，将我们的网络输出的矩阵转为双随机矩阵。

2.2. Sinkhorn 算法

Sinkhorn 算法，也称为 Sinkhorn-Knopp 算法，是一种用于计算正规化的矩阵的算法。该算法的目标是通过行和列的缩放操作，将一个给定的非负矩阵变换为一个双重随机矩阵（即行和列之和均为 1 的矩阵）。Sinkhorn 算法的伪代码描述如下所示：

Sinkhorn 算法通过交替控制行和列的缩放因子直至收敛来生成双随机矩阵作为我们网络输出的预测概率。对于行和列的归一化操作，每次通过除以行和或列和来完成。即对于 $n \times n$ 的矩阵 M，行归一化的操作是

$$M_{ij} \leftarrow \frac{M_{ij}}{\sum_{k=1}^{n} M_{ik}}$$

而列归一化的操作是

$$M_{ij} \leftarrow \frac{M_{ij}}{\sum_{k=1}^{n} M_{kj}}$$

Algorithm 1 Sinkhorn 算法

Require: 非负矩阵 $\mathbf{A} \in \mathbb{R}^{n \times m}$, 容差 $\epsilon > 0$
Ensure: 双重随机矩阵 **B**
初始化矩阵 $\mathbf{B} \leftarrow \mathbf{A}$
while 未收敛 **do**
  for 每一行 i **do**
    归一化行：$\mathbf{B}_{i,:} \leftarrow \mathbf{B}_{i,:} / \sum_{j=1}^{m} \mathbf{B}_{i,j}$
  end for
  for 每一列 j **do**
    归一化列：$\mathbf{B}_{:,j} \leftarrow \mathbf{B}_{:,j} / \sum_{i=1}^{n} \mathbf{B}_{i,j}$
  end for
  if 行和列的和均达到预定的容差范围内的和为 1 **then**
    收敛，退出循环
  end if
end while
return B

按照上面的方式一直迭代更新直到收敛或者达到最大迭代次数，就可以得到一个行和列之和均为 1 的双随机矩阵。它可以作为我们神经网络的输出预测值。

3. 实验结果分析

3.1. 评价指标

在这个问题上，我们引入几个常见的评价指标。为了评价模型预测的准确率，我们引入了总体准确率和分块准确率。总体准确率指的是对于将图片中所有子图全部排列正确的数量占总图片样本数量的比例。若总共有 n 个图片样本，每个图片被切分成 m 个子图，则总体准确率可以由下面的公式描述：

$$\tau_T = \frac{1}{n} \sum_{i=1}^{n} \mathbf{1}[P_i = Q_i]$$

在上面的式子中，P_i 代表第 i 张图片的标签，而 Q_i 代表第 i 张图片对于 m 张子图位置的预测值。同时我们还引入了分块准确率。分块准确率只描述每张子图的分类准确率，也就是说，只统计子图被正确定位的数量占子图总数的比例。分块准

确率可以用下面的公式描述：

$$\tau_F = \frac{1}{mn}\sum_{i=1}^{n}\sum_{j=1}^{m}\mathbf{1}[P_{i,j} = Q_{i,j}]$$

在上面的式子中，$P_{i,j}$ 和 $Q_{i,j}$ 分别代表第 i 张图片的第 j 张子图的标签和预测值。

3.2. 实验结果

3.2.1. 2×2 子图

利用上面介绍的构造数据集的方式和网络结构进行实验。先选取最简单的 2×2 的图片切分方式，将生成的双随机矩阵和标签的交叉熵损失作为损失函数，每个 batch 包含 1024 张图片，以学习率为 0.01 训练 30 个 epoch。我们的预测值是输出的矩阵中概率最大的位置。在训练过程中，分别观察训练集和测试集上的 loss 变化，做出如下图所示的曲线：

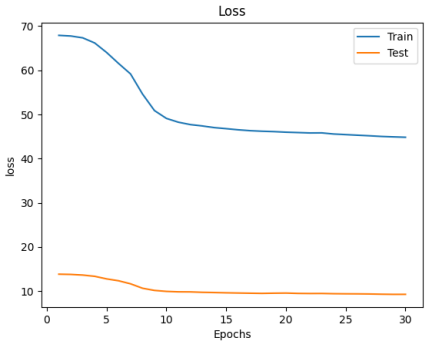

图 4: *loss* 变化图

从上面的图中可以看出，在经过经过 30 个 epoch 后，loss 基本不变，模型达到收敛。每一个训练轮次后，分别测试模型在训练集和测试集上的总体准确率和分块准确率，可以得到如下所示的变化曲线：

而分块准确率的变化曲线如下图所示：

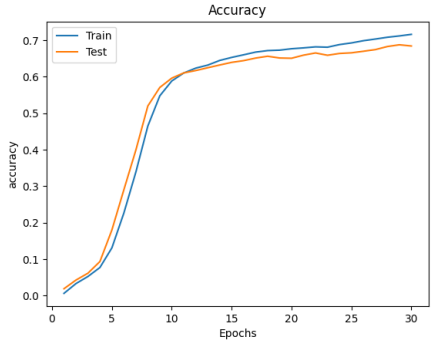

图 5: 总体准确率变化图

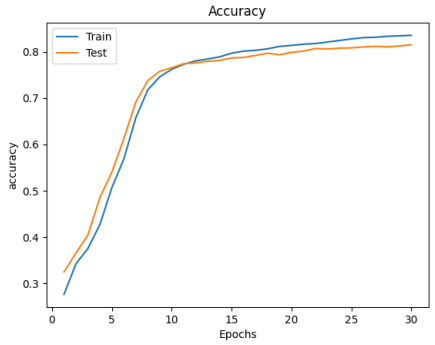

图 6: 分块准确率变化图

将训练好的模型应用在测试集上，测试它的总体准确率和分块准确率如下表所示：

表 1: 准确率指标

	总体准确率	**分块准确率**
训练集	71.6%	83.5%
测试集	68.4%	81.5%

可见我们的模型在测试集上取得了不错的预测效果。并且在测试集和训练集上的总体准确率和分块准确率差距不大，泛化性能较好。

3.2.2. 4×2 子图

同时我们还针对 4×2 的子图进行了测试。我们以每个 batch500 张图片，学习率为 0.01 训练 30 个 epoch。下面是 loss 随训练轮次变化的曲线图：

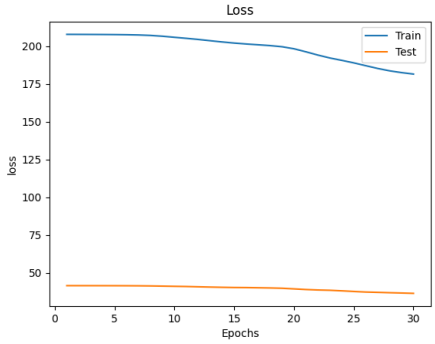

图 7: *loss* 变化图

同时做出了在训练集和测试集上分块准确率随训练轮次变化的曲线图，如下图所示：

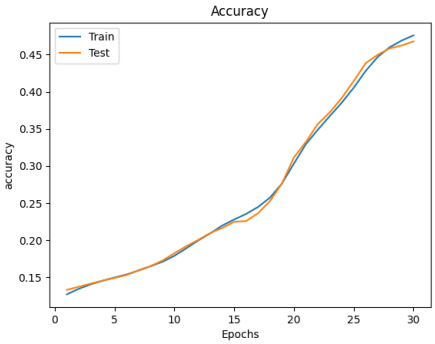

图 8: *loss* 变化图

将训练好的模型应用在测试集上，测试它的总体准确率和分块准确率如下表所示：

表 2: 准确率指标

	总体准确率	**分块准确率**
训练集	8.71%	47.6%
测试集	7.67%	46.8%

可见在 4×2 的子图任务上，我们的模型的总体准确率并不高，预测结果并不理想。而这可能是因为在 4×2 的子图中，每个子图的形状为 8×16，每张图片包含的像素太少，难以完全提取出它的图像特征。因此一个可能的改进方法是做数据增强，将每张图片的像素量增大，这样才能更好的提取出特征。并且由于计算资源有限，我们的网络参数量太小，可以增大模型的参数量，利用更多的网络层参数，提高预测的准确率。