深度学习大作业：拼图问题

521030910376 应之未

在本次大作业中，我主要参考了 jittor 官方文档库和作业描述中提供的参考论文。本次大作业的代码可以在本链接中找到，文件夹中包含两个文件，“data_generate.py” 文件包含生成数据集部分的代码，“hw.py” 文件包含网络设计、训练和测试部分的代码。

1. 问题背景及任务描述

图像数据中往往包含丰富的时间和空间结构，例如一个物品周围的背景信息通常能为这个物体的识别提供一定的线索。再举一个更加具体的例子，当我们预测一个人的运动轨迹时，一种可行的方法是我们识别出环境的特征，得到地平面的相对坐标值，用来校准人体的脚部关键点的位置。因此，对于图像数据中的时空结构的提取是一个有价值的问题。

在本次大作业中，我们主要关注图像的连续性这一空间结构特点。当一张图像被裁剪成小块时，能不能训练出一个模型通过对每个小块关键特征的提取使得其能将图像重新排序成为合适的顺序。

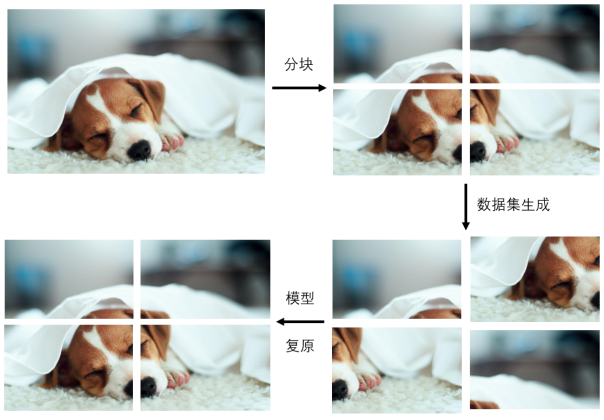

图 1: 任务图示

在本次作业中，我主要参考了相关论文中提出的的 Deep PermNet 模型 [1]。Deep PermNet 模型使用卷积神经网络来解决视觉排列问题，通过对小块图像使用 CNN 进行特征提取后经过一系列网络输出一个对图像排序的矩阵。在生成排序矩阵时，该模型使用了 Sinkhorn 层，利用 Sinkhorn 迭代将预测得到的原矩阵转换为归一化后的双随机矩阵。由于这些矩阵是离散置换矩阵的连续近似，因此允许通过反向传播进行有效的学习。

2. 数据集生成

本次大作业的数据集需要自己根据作业描述利用已有的 CIFAR10 数据集生成，因此，在第一节中我将详细阐述我的数据集生成方式。

利用 jittor 已经集成好的 CIFAR10 类，我生成数据集的伪代码如下：

```
# 利用CIFAR10类生成训练集和测试集原始图像，并设置batch_size
train_set = CIFAR10()
test_set = CIFAR10(train=False)
train_set.set_attrs(batch_size)
test_set.set_attrs(batch_size)

# 生成训练集图像和标签
for imgs, labs in train_set:
    # 产生[1,2,3,4]的随机排列，用于打乱图片顺序
    list = [1,2,3,4]
    random.shuffle(list)
    img = None
    label = []
    for i in range(4):
        if (list[i] == 1):
            将(imgs[:, 0:16, 0:16, :]/255.0).reshape(batch_size, 16, 16, 3)存入img中
            将[1,0,0,0]存入label中
        if (list[i] == 2):
            将(imgs[:, 0:16, 16:32, :]/255.0).reshape(batch_size, 16, 16, 3)存入img中
            将[0,1,0,0]存入label中
        if (list[i] == 3):
            将(imgs[:, 16:32, 0:16, :]/255.0).reshape(batch_size, 16, 16, 3)存入img中
            将[0,0,1,0]存入label中
        if (list[i] == 4):
            将(imgs[:, 16:32, 16:32, :]/255.0).
```

```
    reshape(batch_size, 16, 16, 3)存入img中
            将[0,0,0,1]存入label中
    images_train.append(image)
    labels_train.append(label)

# 保存训练集图像和标签
images_train = np.array(images_train).transpose
    (0,1,4,2,3)
np.save('images_train.npy',images_train)
labels_train = np.array(labels_train)
np.save('labels_train.npy',labels_train)

# 生成测试集图像和标签，内容与训练集一致，不再重复
for imgs, labs in test_set:
    ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ...
    images_test.append(image)
    labels_test.append(label)

# 保存测试集图像和标签
images_test = np.array(images_test).transpose
    (0,1,4,2,3)
np.save('images_test.npy',images_test)
labels_test = np.array(labels_test)
np.save('labels_test.npy',labels_test)
```

3. CNN 网络

本次作业中，我参考了作业描述中的说明以及提供的相关论文 DeepPermNet: Visual Permutation Learning，设计了以下 CNN 网络：

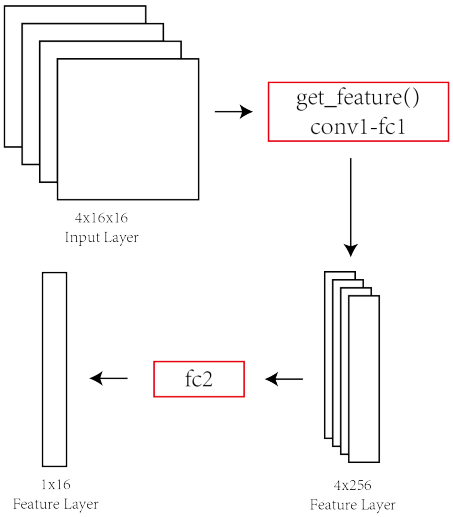

图 2: 网络结构

其中，get_feature() 部分的卷积神经网络结构如下图：

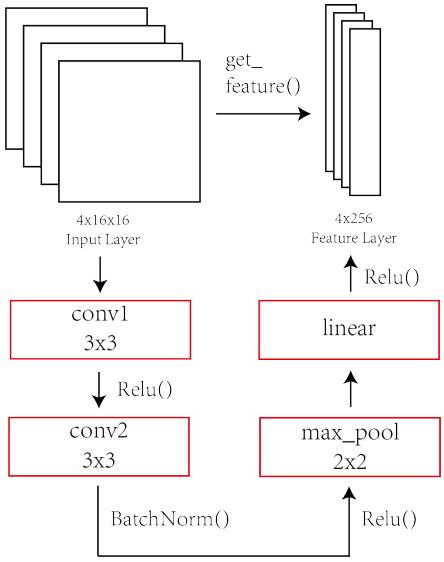

图 3: *get_feature()* 网络结构

网络搭建的代码如下：

```
class Model (Module):
    def __init__ (self):
        super (Model, self).__init__()
        self.conv1 = nn.Conv(3, 32, 3)
        self.conv2 = nn.Conv(32, 64, 3)
        self.bn = nn.BatchNorm(64)
        self.max_pool = nn.Pool (2, 2)
        self.relu = nn.Relu()
        self.fc1 = nn.Linear (64*6*6, 256)
        self.fc2 = nn.Linear (1024, 16)

    def get_feature (self, x):
        x = self.conv1 (x)
        x = self.relu (x)
        x = self.conv2 (x)
        x = self.bn (x)
        x = self.relu (x)
        x = self.max_pool (x)
        x = jt.reshape (x, [x.shape[0], -1])
        x = self.fc1 (x)
        return x

    def execute (self, x) :
        x = self.get_feature(x)
        x = x.reshape(-1)
        x = self.fc2(x)
        x = x.reshape(4,4)
        x = pygm.sinkhorn(x)
        x = x.reshape(-1)
        return x
```

其中，get_feature 函数用于对输入的图片提取特征。将 N 张子图的特征通过 get_feature 函

数分别提取后，把得到的总特征 flatten 成一维，再通过一个全连接层得到一个 N^2 维的数组。将得到的 N^2 维数组 reshape 成 $N \times N$ 的矩阵，通过 sinkhorn 函数将其归一化，最后重新 flatten 成 N^2 维的数组，即为网络的输出。

输入层接收的输入为 4 张乱序的 16×16 大小的子图，输出一个 16 维数组。16 维数组还需要 reshape 成 4×4 的矩阵之后用 sinkhorn() 函数进行归一化，然后又展平成一个 16 维的数组，即为最终的输出。

4. 训练

训练相关参数如下表：

表 1: 训练相关参数

参数名	参数值
learning_rate	0.01
num_epochs	8
momentum	0.9
weight_decay	1e-4

训练部分的伪代码如下：

```
初始化模型 model = Model()
初始化损失函数 lossfunc = nn.MSELoss()
初始化优化器 optimizer = nn.SGD(model.parameters()
    , learning_rate, momentum, weight_decay)

设置模型为训练模式 model.train()
for epoch in range(epochs):
    for i in range(训练集长度):
        outputs = model(images_train[i])
        targets = np.array(labels_train[i])
        loss = lossfunc.execute(outputs, targets)
        optimizer.step (loss)

保存模型到路径'./cifar10_model.pkl'
```

5. 测试

测试部分的伪代码如下：

```
设置模型为测试模式，不再更新参数 model.eval()
for i in range(测试集长度):
    outputs = model(images_test[i])
    targets = np.array(labels_test[i])
    将outputs转化为预测结果pred
    将targets转化为正确结果gt
    if (gt == pred):
        acc += 1
    num += 1
输出测试集的准确率
```

调参的过程由于并没有做一些有技术含量的事情，只是单纯的尝试不同参数，因此予以省略。最终实验尝试出的较优的训练相关参数即为第四节训练中所呈现的相关参数。经过测试，该模型经过训练后在测试集上取得了 0.5506 的 accuracy。

从概率上说，四张子图的排序方式有 $4! = 24$ 种，因此，针对本问题的 random baseline 为 $1/24 \approx 0.042$。结合实验结果可以发现，这样的空间结构特征的连续性是 CNN 网络可学的。

6. 问题拓展与延伸

在完成大作业的过程中，我发现关于本问题还有一些可以拓展的部分，由于时间有限未能进行研究，现将其罗列如下：

- 当划分子图的数量变大，子图变小时，能否找到更好的提取特征网络来提高准确率。
- 当子图的大小不一致，或者形状不统一时，怎么找到合理的处理方式。
- ……

7. 总结与感受

在本次作业中，我感受到了 jittor 和 torch 的两点不同：在 torch 中需要注意一个数据是放在 cpu 上还是 gpu 上，但是在 jittor 中，这一部分的任务不需要使用者担心；两者在损失函数反向传播梯度时的代码结构也有明显不同。

8. References

[1] R. S. Cruz, B. Fernando, A. Cherian, S. Gould, Deeppermnet: Visual permutation learning (2017). arXiv:1704.02729.