深度学习作业三报告

520030910342 柳纪宇

1. 概述

在本次实验中，我在 Jittor 框架下搭建了 DeepPermNet 模型实现了图片拼接任务。即将一张完整的图片切分成多个子图并打乱作为输入，通过该模型我们根据这些乱序数据还原了子图的正确排列位置。

此外，我还尝试了使用图片拼接模型作为图片分类问题的一种预训练方式以提高图片分类任务的模型准确率。

2. 实验过程

2.1. 数据准备

本次实验中使用的数据集为 CIFAR-10，在调用作业二的数据集接口的基础上，我对返回的数据和标签做了相应的改动。

首先，我以图片宽高的中点为分界将 3x32x32 的原图片拆分成四张 3x16x16 的子图，依次添加到列表 image_list 之中。之后，我将它们的位置信息（0，1，2，3）添加到列表 position_list 中。图片切分和标签生成的可视化效果可见图 1。

将上述两个列表以相同的顺序进行 shuffle 操作，得到打乱了的图片列表和位置列表，将它们转换成 jittor.Var 类型的数组作为数据和标签返回。

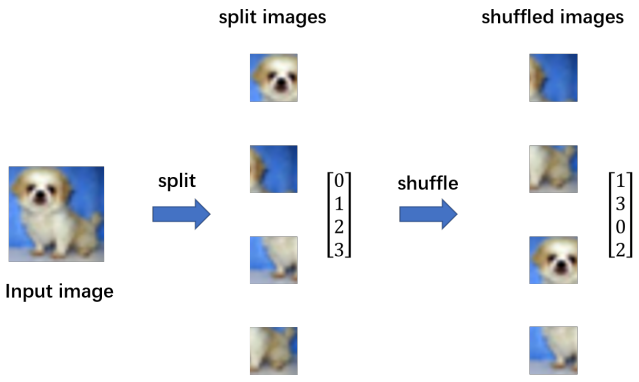

图 1: 图片切分及标签生成

2.2. 模型准备

本次拼图实验中，我参考了 GitHub 项目 optimal_transport_problem [1]中的 DeepPermNet 设计，以此为基础实现了自己的 DeepPermNet 模型。

该模型可以进一步拆分成特征提取器和排列提取器两部分。我将在接下来的三个小节中依次介绍它们的结构和功能。

模型结构的整体可视化效果可见图 2。

2.2.1. 特征提取器

特征提取器的作用为提取输入图片的特征向量。本模型使用的特征提取器为一个双层 CNN 模型。其中 CNN 的每一个隐层都由依次排列的卷积层、BatchNorm 层、Relu 层和 Pool 层组成。将 3x16x16 大小的数组输入该特征提取器后将会得到一个 4x4x32 的特征数组，接着对其进行 Flatten 操作，得到 feature_size 维的特征向量作为输出。

2.2.2. 多头特征提取器

多头特征提取器的作用可以看作是同时用特征提取器提取了 N 个子图的特征向量并将其合并成一个 N x feature_size 大小的特征矩阵。其具体实现方法为，将输入数组的 head_size 维和 batch_size 维进行合并后输入特征提取器，输出结果为一个 (head_size * batch_size) x feature_size 的特征矩阵。为保持前后 batch_size 维度的一致性，将其 reshape 为一个 batch_size x (head_size * feature_size) 的特征矩阵并输出。

[1] https://github.com/Godofnothing/optimal_transpor_problem/tree/1916077191397e35857ffe5a76113a4f46179b01

2.2.3. 排列提取器

排列提取器的作用为将多头特征提取器输出的特征矩阵转变为具有排列顺序意义的排列阵。具体实现方法为将特征矩阵依次通过全连接层 1、Relu 层和全连接层 2，将输出结果 reshape 成一个 batch_size x head_size x head_size 的矩阵并通过 Sinkhorn 算法使其变为行列和固定为 1 的双随机矩阵作为预测值输出。

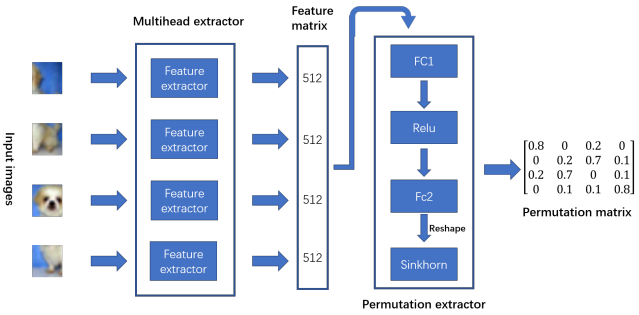

图 2: *DeepPermNet* 模型结构示意图

2.3. 模型训练及测试

2.3.1. 超参数选取

本次实验使用 SGD 优化器进行模型优化；学习率选取为 0.01；训练轮数选取为 50；损失函数选取为交叉熵函数；测试集和训练集的 batch_size 都选取为 16；特征向量的维数 feature_size 选取为 512；多头特征提取器接口个数（即子图个数）head_size 选取为 4。

2.3.2. 模型训练与测试

模型训练的方式与作业 2 中相似。将由四张子图组成的数组输入模型得到位置矩阵，将其与标签值计算交叉熵误差并进行梯度回传更新参数。每 100 次迭代输出一次平均误差，每个 epoch 记录一次总误差值。

模型测试的思路主要在于将模型输出的位置矩阵转换为与标签类型相同的位置向量进行准确率计算。具体实现方式为对输出矩阵的第二维（即每一行）进行 argmax 操作，得到矩阵中每行最大值的所在位置，将其作为表示排序位置的向量与标签值相比较计算测试准确率。每 1000 次迭代输出一次平均准确率，每个 epoch 记录一次平均准确率。

模型测试的实现流程如图 3 所示。

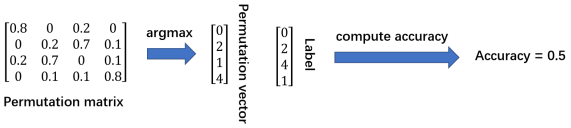

图 3: 模型测试示意图

2.3.3. 训练结果

训练 50 个 epoch 后的模型参数保存在'./trained_model/deeppernet_model.pkl' 中。模型训练过程中的训练误差曲线和测试准确率曲线见图 4、图 5。训练过程中的最佳测试准确率为 0.83965。

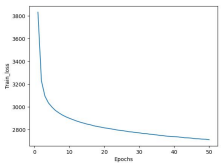

图 4: 拼图模型训练误差

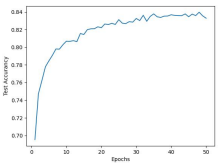

图 5: 拼图模型测试准确率

2.4. 预训练辅助图片分类任务

在这个部分中，我尝试了将以上的图片拼接操作作为一种预训练方式来辅助作业二中的图片分类任务。

2.4.1. 模型定义与参数加载

为了与作业二中的模型效果进行比较，除了处于算力限制考虑将训练轮数从 50 减少到了 25（先前实验表明模型在 20 个 epoch 左右就达到收敛，而本次实验的结果也印证了减少训练轮数并未对最终结果造成明显影响）之外，本次图片分类保留了作业二中使用过的双层 CNN 模型和相关超参数。由于 DeepPermNet 中的特征提取器结构与去掉全连接层的 CNN 模型相同，所以将

DeepPermNet 中的参数加载到分类器 CNN 模型中，模型的参数就不再是随机初始化生成的，而是经过拼图模型预训练的结果。

2.4.2. 训练结果

训练 25 个 epoch 后的模型参数保存在'./trained_model/classifier_model.pkl' 中。模型训练过程中的训练误差曲线和测试准确率曲线保存见图 6、图 7。训练过程中的最佳测试准确率为 0.6878。相较作业二中使用 CNN 模型达到的最佳准确率 0.5766 有了较大的提升，其效果甚至超过了 jittor 自带的预训练模型。

此外，在将预训练后的模型训练曲线同作业 2 中的曲线（见图 8、图 9）进行比较时，我注意到相较随机初始化参数的训练过程，预训练模型的训练过程更加平稳，抖动和干扰也变得更小了。

这说明拼图模型作为分类模型的初始化方式不仅有效提高了模型准确率。也提高了模型的抗扰动能力。

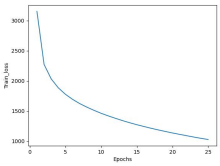

图 6: 分类模型训练误差

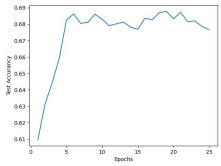

图 7: 分类模型测试准确率

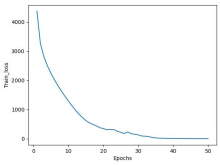

图 8: 原分类模型训练误差

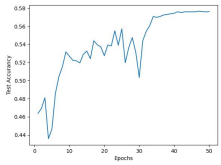

图 9: 原分类模型测试准确率

3. 总结

在本次实验中，我参考已有的 DeepPermNet 模型搭建了自己的深度神经网络，实现了返回乱序子图的正确顺序的效果，其中模型的最佳准确率达到了 0.83965。在此之外，我利用了 DeepPermNet 中的特征提取器与图片分类 CNN 的结构相似性，将训练后的 DeepPermNet 参数作为 CNN 的初始化参数以达到预训练的效果。最终在模型结构和超参数都相同的情况下，预训练过程将双层 CNN 模型在 CIFAR-10 数据集上的准确率由 0.5766 提升到了 0.6878。