COMP7250 Machine Learning Course Project

Topic - 3: Bag of tricks for effective training

Name: Zheng Yifei Student ID: 22432035

ResNet-9

ResNet-9 is a convolutional neural network that has been extensively trained on CIFAR-10 dataset. I reference its architecture has been derived from a smaller version of the model presented in the GitHub(GitHub - Moddy2024/ResNet-9: Designed a smaller architecture implemented from the paper Deep Residual Learning for Image Recognition and achieved 93.65% accuracy.). My primary objective in this exercise has been to ensure that there is a fair balance between train-time cost and the validation accuracy. Using ReLU activations across all convolution layers, alongside the use of a 20% dropout rate. I have implemented these on the Adam optimizer while keeping the learning rate steady at 0.01.

Dependencies

- Pytorch
- Matplotlib
- PIL
- Numpy
- Torchvision
- Torchinfo

Once you have these depdencencies installed, you can run the file.

Training and Validation Image Statistics

The dataset used to train the model is CIFAR-10. The CIFAR-10 dataset consists of 60,000 32x32 color training images and 10,000 test images. Each image is labeled with one of 10 classes: airplane, automobile, bird, cat, deer, dog, frog, horse, ship, and truck. There are 6,000 images of each class in the training set, and 1,000 images of each class in the test set. CIFAR-10 is a popular choice for benchmarking because it is a well-defined and widely-used dataset, and the images are small enough that it is possible to train relatively large models on a single machine.

Dataset

The CIFAR-10 dataset can be downloaded from CIFAR-10 and CIFAR-100 datasets . It can also be downloaded from PyTorch Datasets.

```
from torchvision import datasets

cifar10 = datasets.CIFAR10(root='../data',download = True)
```

Common Problem

If there is no CUDA(Error like graph1), delete `model.cuda()` from function `fit_one_cycle`, and then restrat again.

```
AssertionError: Torch not compiled with CUDA enabled
```

```
def fit_one_cycle(method,epochs,model,train_loader,val_loader,weight_decay=0,
        grad_clip=None,opt_func=torch.optim.Adam):
    history = []
    Tlist = []
    optimizer = opt_func
    total_train_time = 0
    for epoch in (range(epochs)):
        # Training Phase
        model.cuda()
        model.train()
```