11785 HW2P2

Name: Xinkai Chen

AndrewID: xinkaic

Code: 11785_hw2p2_xinkaic.ipynb

Steps:

1. Download data and unzip them
2. Import libraries and run on GPU
3. Convert images to tensors, define dataloaders and load training data and dev data
- Used the transform ToTensor() from torchvision
- Data Augmentation: I used some built-in torchvision transforms: Horizontal Flips, Rotations, and Color Jitters
- Reads validation and test data as normal (only ToTensor)
- Dataloader has batch size of 200
4. Define network
- I tried different things:
- Baseline (as defined in Piazza)
- Deeper & Wider VGG-19: I made layers wider (max being 2048); after the 19 layers, add a few more layers to make output channels 512
- Modified ResNet50: I basically implemented the ResNet50, only changing the first layer to have a smaller stride and kernel size, so that more information is kept.
- It turns out that ResNet indeed works better. I've also tried DenseNet after the submission, it can learn quite well but ResNet is good too.
5. Define training parameters
- Criterion: CrossEntropyLoss() (I've tried CenterLoss and several weights attached to it, but it wasn't working very well for me so I didn't include that eventually; but will learn more about it)
- Learning rate: starting with 0.15, *0.85 after each epoch
- Optimizer: SGD with momentum of 0.9 and weight decay of 5e-5
6. Training
- The accuracy reaches 0.7+ after around 15 epochs
- Validate after each epoch
7. Evaluate results and tune hyperparameters
- I tried different networks and tried different losses and tries on the verification validation set. It was very close to the actual score on the leadeboard
8. When the val accuracy is okay, load the test set and predict on test set
9. Generate submission.csv and submit to Kaggle