11785 HW3P2

Name: Xinkai Chen

AndrewID: xinkaic

Code: 11785_hw3p2_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
- Dataset is pretty straightforward; in the collate_fn I padded inputs and labels together. I used batch_first on labels because they will be passed to CTCLoss.
- Dataloader has batch size of 64
4. Define network
- I tried different things:
- Baseline (as defined in Piazza): that gave me about a 11-12 distance
- Added CNN layer and extra Linear layer, also expanded the LSTM layers: I read a paper that used Conv – LSTM – FC, so I used a Conv1d layer with a small kernel size to preserve more features; I made the LSTM layers wider by using 512 hidden units; also deeper by using 4 stacked bi-LSTM layers; added another Linear layer before the output layer.
- I tried using more Conv layers but that didn't work well, so I only kept one layer. I discussed with classmates after the deadline, it turns out that using pooling/downsampling can be helpful.
- I didn't try some regularization tricks on LSTM when I reached distance < 8; but also during the discussion, Dropout can be helpful, and I will definitely look into those when doing HW4.
5. Define training parameters
- Criterion: CTCLoss(), I struggled with batch_first at first but then managed correct it; I read the documentation carefully to send in the correct inputs.
- Learning rate: starting with 0.001, *0.85 after each epoch (I made step_size larger when the training was just started, and then decrease learning rate more frequently as the loss decreases more)
- Optimizer: Adam with weight decay of 5e-6 and the learning rate above
6. Training
- The distance reaches the A cut-off after around 25 epochs
- Validate after each epoch
7. Evaluate results and tune hyperparameters
- I tried different networks and tried different hyperparameters. And then I evaluate the model on the validation set. Also I print out the outputs to see if my model generation makes sense.
8. When the val loss & distance is okay, load the test set and predict on test set.
9. Generate submission.csv and submit to Kaggle.