

Assignment #3: Pytorch CIFAR-10 Image Classification

Due Mar 10 by 11:59pm **Points** 40 **Submitting** a file upload **File Types** zip and pdf

###IMPORTANT NOTES###

This instruction will mainly based on Pelican servers with bash shell and Python 2.x, please feel free to use any resources you have.

If you want to change your default shell to bash, you can do so at:

https://secure.engr.oregonstate.edu:8000/teach.php?type=want_auth

(https://secure.engr.oregonstate.edu:8000/teach.php?type=want_auth)

###Login Server###

Login one of the following servers:

pelican01.eecs.oregonstate.edu

pelican02.eecs.oregonstate.edu

pelican03.eecs.oregonstate.edu

pelican04.eecs.oregonstate.edu

For example, if you ONID is XXXX, you can login through ssh by typing following command in your terminal:

ssh XXX@pelican01.eecs.oregonstate.edu

For Windows users, you can choose to use either MobaXterm or Linux Subsystem (for Windows 10 only).

Once login, you can check whether GPU resources are available in this server using command:

\$:nvidia-smi

###Install Python Virtualenv###

Create a Python virtual environment:

\$ ~:virtualenv --system-site-packages pytorch

Activate your virtualenv:

\$ ~:source ~/pytorch/bin/activate

If everything is correct, you will see there will be a "(pytorch)" at the beginning of your command line

\$(pytorch) ~:

You will install all your Python packages related to assignment 3 in this python virtual environment

If you want to exit, just type:

\$(pytorch) ~:deactivate

###Install Pytorch###

Make sure your Python virtual environment is activated

Upgrade your pip:

pip install --upgrade pip


Install Pytorch and Torchvision package:

```
pip install http://download.pytorch.org/whl/cu80/torch-0.3.0.post4-cp27-cp27mu-linux_x86_64.whl
```

```
pip install torchvision
```

After installation, you should be able to import Pytorch in your python and `torch.cuda.device_count()` should return 2L

###Train Your Network###

Download [cifar10_pytorch.py](#)  to your server, and run following command:

```
python cifar10_pytorch.py
```

If you are using pelican and you want your code to run on a specific GPU, you can add a variable like below:

```
CUDA_VISIBLE_DEVICES=1 python cifar10_pytorch.py
```

Above command will only allow program to use GPU 1(the second GPU on pelican). It should take less than 1 minutes to train 500 steps if you are using GPU instead of CPU.

###Assignment Request###

Now open `cifar10_pytorch.py` and briefly review what's inside, make sure you understand each line of code, and finish following tasks:

1) Add a batch normalization layer after the first fully-connected layer(fc1) (10 points).

Save the model after training(Checkout our tutorial on how to save your model).

Becareful that batch normalization layer performs differently between training and evalation process, make sure you understand how to convert your model between training mode and evaluation mode(you can find hints in my code).

Observe the difference of final training/testing accuracy with/without batch normalization layer.

2) Modify our model by adding another fully connected layer with 512 nodes at the second-to-last layer (before the fc2 layer) (10 points).

Apply the model weights you saved at step 1 to initialize to the new model(only up to fc2 layer since after that all layers are newly created) before training. Train and save the model (Hint: check the end of the assignment description to see how to partially restore weights from a pretrained weights file).

3) Try to use an adaptive schedule to tune the learning rate, you can choose from RMSprop, Adagrad and Adam (Hint: you don't need to implement any of these, look at Pytorch documentation please) (10 points).

4) Try to tune your network in two other ways (10 points) (e.g. add/remove a layer, change the activation function, add/remove regularizer, change the number of hidden units, more batch normalization layers) not described in the previous four. You can start from random initialization or previous results as you wish.

For each of the settings 1) - 4), please submit a PDF report your training loss, training accuracy, validation loss and validation accuracy. Draw 2 figures for each of the settings 1) - 4) (2 figures for each different tuning in (4)) with the x-axis being the epoch number, and y-axis being the loss/accuracy, use 2 different lines in the same figure to represent training loss/validation loss, and training accuracy/validation accuracy.

Name your file "firstname_lastname_hw3.pdf". Submit this pdf file on Canvas.

###Hint for Partially Restore Weights###

In Pytorch trained weights are saved as *.pth file, you can load these files into a dictionary with `torch.load('YOUR_FILE_NAME.pth')` function. If you try to print the keys in this dictionary, you will find they are the variable name you used when you create your model.

So you can load your pretrained weights into a dictionary 'pretrained_dict', then load your new model's weights into another dictionary 'model_dict' with method `net.state_dict()`. Then update weights in 'model_dict' only when keys in model_dict are also in pretrained_dict. Once you have updated model_dict, you can assign the values in this dictionary to your new model using `net.load_state_dict(model_dict)` method.