CMSC5743 Lab 05

Introduction to Distiller

1 Intro to Pruning of Distiller

Steps

- Install Distiller, create and activate a virtual environment `env`
- Make sure that you're at the root directory of `Distiller`
- Define your own model:
  - Add your model into `./distiller/models/cifar10`
  - Register your model at the end of `./distiller/models/cifar10/__init__.py`
- Step into `./examples/classifier_compression`
- Train your model from scratch: run "`python3 compress_classifier.py --arch cmsc5743_cifar ../../../data.cifar10 -p 30 -j=1 --lr=0.01 --epochs 2`"
- Check your log file to locate the checkpoint file
- Evaluate your model: run "`python3 compress_classifier.py --arch cmsc5743_cifar ../data.cifar10 -p=50 --resume-from=XXXX --evaluate`", here `XXX` is your checkpoint file
- Define your pruning algorithm:
  - Add your algorithm into `./distiller/pruning`
  - Register your algorithm in `./distiller/pruning/__init__.py`
- Prune your model:
  - Load the stored checkpoint file to check the weights, an example code is in `./Lab05-code/checkname.py`; or you can get them by evaluating the model
  - Add your own pruning configuration file into `./examples/sensitivity-pruning`
  - Run "`python3 compress_classifier.py --arch=cmsc5743_cifar ../../../data.cifar10 -p=50 --lr=0.1 --epochs=2 --batch=128 --compress =../sensitivity-pruning/cmsc5743_cifar.schedule_sensitivity.yaml --gpu=0 -j=1 --deterministic`" to prune the model

2 Sample Code

- check the model weights: `./Lab05-code/checkname.py`
- Define your model:

  - model file: `./Lab05-code/cmsc5743_cifar.py`
  - register your model: "`from .cmsc5743_cifar import *`"
- Define your pruning algorithm:
  - algorithm file `./Lab05-code/cmsc5743_pruner.py`
  - register your algorithm: "`from .cmsc5743_pruner import CMSC5743Pruner`"
- pruning schedule file: `./Lab05-code/cmsc5743_cifar.schedule_sensitivity.yaml`
- scripts: `./Lab05-code/lab05.sh`

3 Assignment

Q1 Prune the squeezenet1_1 (pretrained model in Distiller)

- Dataset: ImageNet (Tiny ImageNet)
- Use group dependency type `Leader` with at least one dependency group
- Use two `group types`: `Channels` and `Filters`.
- Use two `pruning algorithms`: `Automated Gradual Pruner (AGP)` and `sensitivity pruning`
- You do not need to cover all of these types or algorithms in a single YAML file, in other words, you can implement these algorithms in several configuration YAML files.

Q2 Learn to quantize mobilenet (pretrained model in Distiller)

- Dataset: ImageNet (Tiny ImageNet)
- Quantize the model via 8-bit post-training linear quantizer.

Useful Materials:

- You can use Tiny-ImageNet
- PyTorch models
- Netron
- named_parameters()
- A bug occured when resuming from the store model