ME 592X: Data Analytics and Machine Learning for Cyber-Physical Systems

Homework 4

Homework Assigned on April 5, 2022 Homework Due on: April 18, 2022
One submission per group

Motivation

This homework is to provide an experience of using advanced deep learning architectures applied to datasets relevant to each theme group.

General Instructions

The dataset and problems for each group are slightly different, but the motivation remains the same. Following are some instructions for all the theme groups. Specific instructions for each group are provided in the relevant sections.

Expected Outcome

1. Code-base pushed to git.
2. A short report with results and discussion should be submitted for each group.

Suggestion: Network Modeling and Optimization

1. **Network:** There are a lot of sample codes that are available online in PyTorch website and also in GitHub. By referring to those, you should know that building a model is easier than you thought!

2. **Optimization:** This is usually the tricky part. Some of the things that you can probably do to optimize the model is, changing learning rate (some common ones are 1e-3, 3e-4), input timesteps, optimizer, batch size etc.

Again, remember, the default parameter values don't necessarily provide the best result! Tune some parameters, and present the best result that you can get!

Agriculture Theme
Image Analytics Theme

For this assignment, we are going to train a object detection model (RetinaNet) for wheat detection application. You can read more about this challenge here at https://www.kaggle.com/competitions/global-wheat-detection. The dataset can downloaded from https://www.kaggle.com/competitions/global-wheat-detection/data. The main goal of this assignment is to get familiarized with training an RetinaNet, optimizing it and understanding the main differences with other object detection models. You can read more about RetinaNets at https://arxiv.org/abs/1708.02002

1. Download the dataset using the link provided above, write the code to train a RetinaNet model to perform wheat detection within the data. As an initial step, you may follow the skeleton code at https://www.kaggle.com/code/jainamshah17/gwd-retinanet-pytorch-train/notebook. Report the results and the relevant metrics from training the initial model.

2. Next, try improving the RetinaNet by optimizing the model from various aspects. Some factors you may try are augmenting the data to generate extra data points, trying out different pre-trained model backbones, trying out different optimizers and learning rates as well as modifying the loss functions (for example, what happens if you weight the regression loss higher than the classification loss, etc). Feel free to also modify other factos that you think might be relevant. Discuss the modifications you made and compare the results of these modifications.

3. Finally, include a discussion on your understanding of RetinaNet and how does it differ from other object detection models, such as Fast-RCNN and Faster-RCNN.

Engineering Design Theme

Point clouds are unstructured representations of 3D models that contains the `(X, Y, Z)` coordinates of points representing the surface of the object. In this assignment, the task is to train a deep learning model to detect objects represented by point clouds. For this task, two different implementations of point cloud based object detection models have been provided. The assignment involves getting familiar with the two frameworks and implementing these. The steps required for the assignment are provided below:

1. Read the PointNet arXiv paper for a brief understanding of the implementation and the model. Link: https://arxiv.org/abs/1612.00593
2. Read the PointNet++ arXiv paper. This is the modified implementation of the previous work. Link: https://arxiv.org/abs/1706.02413
3. Implement the PointNet framework using Tensorflow as given in the Github link: https://github.com/charlesq34/pointnet. The datasets used for this are ModelNet40 point cloud version and ShapeNetPart. All the details regarding the dataset and how to implement is provided in the Github link. Try modifying the hyperparameters of the model in the code to achieve a better classification and segmentation result for the two datasets respectively.
4. Repeat 3 for PointNet++ framework as given in the Github link: https://github.com/charlesq34/pointnet2.
5. Submit the trained weights to repository if the size allows you to push it along with the codes for both the frameworks. Also submit a write-up on the difference in the two implementations and report the results comparing the two.

Engineering Systems Health Monitoring Theme

For this assignment, you will explore a dataset comprising hi-speed images from a combustion system which can be downloaded from
https://iastate.box.com/s/kl7upcmlqlewq8zedq1huo9syrtgqpw0. The dataset consists of flatten arrays which you would have to reshape into 100 x 250 images and all images belong to either one of the two classes: stable, unstable. The goal of this assignment is to train a classifier to predict if the combustion system is stable or unstable from the image(s).

Task 1: Train a 2D Convolutional Neural Network for classifying the stable and unstable flames. Try to optimize the hyperparameters to get the best possible classification accuracy. Report your results and observations while tuning the hyperparameters. Keep the relatively stable set of images aside for testing. Report the results for that set.

Task 2: The images in the dataset are sequential in nature. Train a 3D Convolutional Neural Network by forming stacks of images. Each stack is either stable or unstable. Perform experiments changing the number of images (N) in a stack. You can experiment with values of N = 4, 8, 12, 16. Try augmenting the dataset if required (there can be overlapping images between two consecutive stacks). After reporting your observations while maximizing the classification accuracy, report your test results on the relatively stable set.

Robotics 1 Theme

In this assignment, you will be training a deep reinforcement learning agent to solve some classic control problems such as the Lunar Lander problem. In the Lunar Lander problem, the goal of the agent is to learn to safely land the Lunar Lander on the landing pad. The actions space of the lunar lander includes vertical and horizontal thrust.

1. Install OpenAI's gym environments and PFRL's deep reinforcement learning library.
2. Follow PFRL's quickstart guide and train a simple DQN agent to solve the LunarLander problem.
3. Try to modify the code to improve the training of the RL agent. The parameters that you can try modifying are the DQN's architecture (number of hidden layers and hidden nodes), activation functions (ReLU, Tanh, etc), exploration ratio, optimizers (Adam, SGD, RMSProp, etc), replay start size, Q-network update interval and the target Q network update interval. Plot the rewards, average q-values and the loss and comment on which factors improves or hampers the agent's training.

- PFRL's repository https://github.com/pfnet/pfrl
- OpenAI's gym repository https://github.com/openai/gym
- Feel free to use any other RL repositories or packages that are open-sourced besides PFRL

Robotics 2 Theme

In this assignment, you will be training a deep reinforcement learning agent to solve some classic control problems such as the Cartpole and Lunar Lander problem. In the Cartpole problem, the agent's task is to learn the correct amount of horizontal force to apply to the cartpole to maintain it's balance. In the Lunar Lander problem, the goal of the agent is to learn to safely land the Lunar Lander on the landing pad. The actions space of the lunar lander includes vertical and horizontal thrust.

1. Install OpenAI's gym environments and PFRL's deep reinforcement learning library.
2. Follow PFRL's quickstart guide and train a simple DQN agent to solve the Cartpole problem.
3. Follow the same steps above and train another RL agent to solve the LunarLander problem.
4. Try to modify the code to improve the training of the RL agent. The parameters that you can try modifying are the DQN's architecture (number of hidden layers and hidden nodes), activation functions (ReLU, Tanh, etc), exploration ratio, optimizers (Adam, SGD, RMSProp, etc), replay start size, Q-network update interval and the target Q network update interval. Plot the rewards, average q-values and the loss and comment on which factors improves or hampers the agent's training.

- PFRL's repository https://github.com/pfnet/pfrl
- OpenAI's gym repository https://github.com/openai/gym
- Feel free to use any other RL repositories or packages that are open-sourced besides PFRL

Autonomous Vehicles 1 Theme
Autonomous Vehicles 2 Theme

In this assignment, you will be running inference of well-known object detection models such as Mask R-CNN and Faster R-CNN. These deep learning object detection models are based on region proposal convolutional neural networks (R-CNN), you can read through a summary of R-CNNs here.

Mask R-CNN

1. Install Mask R-CNN implementation from Facebook Research.
2. Installation instructions can be found here.
3. If installing on Windows, follow the installation instructions using Miniconda. You might need to install Microsoft Visual Studio C++ Build Tools.
4. Run webcam inference `webcam.py` from `demo` as shown in here. If you have CUDA installed in your computer, you can select GPU for real-time object detection, else select CPU.

Faster R-CNN

1. Install Faster R-CNN implementation from ChainerCV.
2. Installation instructions can be found here.
3. Run `demo.py` from `examples/faster_rcnn` with an image of your choice.

Compare detection performance on both models by feeding in the same image. Test out with several different images such as images trivial to humans vs extreme challenging conditions to humans. Try preloading with different network weights and observe the difference in detection performance (examples above used some pre-trained network weights for inference). Write a short report showing those differences.