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

Homework 3

Homework Assigned on March 23, 2023 Homework Due on: April 6, 2023
One submission per group

Motivation

This homework is to provide an experience of deep learning using simple networks 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 shall be provided in the relevant sections.

1. The final code must be pushed to git before the deadline.
2. Use the discussion board in Canvas in case of any issue.

If groups have technical questions related to their homework, don't hesitate to get in touch with the following individuals:

- Agricultural Analytics: Timi (`ayanlade@iastate.edu`)
- Engineering Image Analytics: Ethan (`edherron@iastate.edu`)
- Robotics: Ryan (`hjy@iastate.edu`)
- Transportation: Zahid (`zahid@iastate.edu`)

Expected Outcome

1. A code pushed in git(bitbucket).
2. A presentation video explaining the problem and results (duration must not be more than 10 mins); we will present this video for some groups in class.

.

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 (try the magic number 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! :)

Agricultural Analytics

In this homework, you would use the similar dataset which you used in the previous assignment about the leaf detection (You can download from `https://iastate.box.com/s/1aejk7t400i2pncbxhhwwu9c5q4ejedu`). Perform the following tasks. A plant can be classified as healthy or diseased based on the color and features in observed in the image. Further, based on the features observed in the leaf, we could categorize into several classes of diseases. The broad classification of the diseases are 6 in number (plus healthy leaves, makes it total of 7).

1. Load all the data provided and split the data into training and testing. Now, using 3-4 different bounding box sizes, construct CNN models which would classify the leaf based on the diseases.

2. Taking the bounding box size into account, perform the following data augmentations on-the-fly(i.e. for every mini batch you would need to generate augmented data). Recommended augmentations are (you can implement more than this):

   (a) Randomly flipping/rotating/mirroring/shearing the image.

   (b) Shifting by random values horizontally or vertically.

   (c) Change the scale of the image and pad.

   (d) Arbitrarily changing the intensity/channels or brightness of the image.

3. Train all the models and perform hyper-parameter optimization.

4. With the newly provided canopy image data containing multiple leaves, using different CNN models learnt till now and device a strategy to automate the annotation of each leaf's classification in the image based on the classification provided by the CNN models trained.

Engineering Image Analysis

For this assignment, you will explore a dataset comprising hi-speed images from a combustion system. The dataset has two classes- stable, unstable (You can download from `https://iastate.box.com/s/1aejk7t400i2pncbxhhwwu9c5q4ejedu`). The data in the .mat file from the link above contains train, test, and validation sets of x's (images) and y's (stable or not stable value). The images are one channel of 250×100 pixels.

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.
2. The images in the dataset are sequential in nature, but too large to use a sequential model. First, train an autoencoder with a small enough bottleneck for a sequential model to operate on. Next, predict the stability of the sequence of input images with a sequential model. The input to the sequential model will be the condensed latents from the pretrained autoencoder. After reporting your observations while maximizing the classification accuracy, report your test results on the relatively stable set.

Your final video should include analysis of the 2D classification model, the autoencoder model, and the sequential classification model on the stable test set.

Robotics

In this assignment, we will be building a simple supervised deep learning model to autonomously guide a robot to explore its environment. We will use Udacity's Rover simulator to collect data and train a Convolutional Neural Network to predict the control signal required to navigate the rover through the unknown terrain without crashing into the cliffs. First, download the Rover simulator from `https://github.com/udacity/RoboND-Rover-Project` and explore the simulator. Instructions on how to run the simulator and log the data can be found on the simulator's screen.

1. Turn on grid mode in the simulator and navigate the Rover around in the simulator to collect a training and testing dataset. Ideally, your training and testing dataset should be collected under nominal navigation conditions. i.e. Not intentionally driving the rover into cliffs.

2. Build and train a Convolutional Neural Network to predict the steering angles and throttle using the images collected in the step above. You may choose to use a classification or regression model for this task and also feel free to use any network architecture that is suitable. If you decide to use a classification model, you need to discretize the control signals into appropriate bins.

3. Run an inference on your testing data using the trained model and provide a few anecdotal examples of the images and the predicted steering angle and throttle controls. Ideally, the model shouldn't be predicting control signals that would cause it to crash into the cliff walls.

4. Collect another dataset with more erratic navigation and perform an ablation study on the effect of the different dataset on the performance of the model. Additionally, conduct another study on the effects of training data size on the performance of the model.

Transportation

In this homework, you will be training a simple network to classify the weather conditions while driving. Berkeley DeepDrive dataset, `https://bdd-data.berkeley.edu/`, contains a large amount of annotated driving data such as 2D bounding boxes, image segmentation, drivable areas and so on. You can read more about the dataset here, `https://arxiv.org/pdf/1805.04687.pdf`.

1. Download and unzip the dataset. (It's a large dataset, so be aware about it). You'll have to register an account and login to download the dataset.
2. Understand how the dataset is managed, such as how each annotations correspond to its image etc. `https://github.com/ucbdrive/bdd-data`. They seperated the dataset by videos, labels, images etc.
3. Build and train a Convolutional Neural Network (CNN) to correctly classify the correct weather attributes for a given image. Feel free to find suitable architectures/networks for inspiration, just make sure you cite where you get it from.
   - The training dataset provided have an imbalanced distribution of certain weather attribute as compared to others. Explore around to find the perfect balance of dataset to feed into your model.
4. Run an inference on the test dataset provided with your best performing model. You can even show how well your model perform on your own driving images (not from BDD).
5. Convert the trained model with weights to a frozen model and save it as a Protocol Buffer. *Read about ONNX.*