Paper 31 Artifact Instructions

Artifact Instructions

0.1 Virtual Machine Setup

Open VirtualBox. Click on File –> Import Appliance and choose "artifact.ova" as the file to import. This will set up the virtual machine in Virtual Box. The provided virtual machine configuration has 2048 MB of memory and 1 CPU allocated to it. Click "Start" to boot the virtual machine. Sign in to User. The password for User is "pass". Open the terminal and navigate to the "~/artifact" directory.

0.2 Artifact Directory Structure

The "experiment.py" file is used to interact with our tool. The sections below contain more information on using the tool. "boxprop_optimized.py" and "lipprop.py" contain code for the box analysis and Lipschitz analysis respectively. The "mnist", "small_cifar", and "large_cifar" folders contain Python scripts invoked by experiment.py to execute our analyses on the corresponding programs. "pretrained_generators" and "pretrained_classifiers" contain the pretrained generative and classifier models (in particular, the learned parameter weights). The "precomputed_results" folder contains the results of each experiment from our paper. The "__pycache__" folder contains files generated by the Numba library for code optimization. The README file contains more information about the artifact directory structure, and this instructions file is also included in the artifact directory.

0.3 Understanding the Command-Line Tool

The “experiment.py” file takes multiple arguments to handle different experiment scenarios. The first argument is the filename of the file containing the generator model. The second argument is the dimension of the latent space of the generator model, which is equivalent to the dimension of the input to the generator model. The third argument is the filename of the file containing the classifier model. The fourth argument is the number of different box sizes, n, that will be tested. The next n arguments are the actual sizes of the boxes. The third-to-last argument is the number of random centers for the boxes. The second-to-last argument is the random seed to generate the random centers for the boxes. The last argument is the program name, and it will be used to name the .csv and .png files. Refer to the section below for examples using “experiment.py”.

0.4 Commands for Running Experiments from the Paper

- "python experiment.py mnist_g.py 100 mnist_f.py 3 0.00001 0.001 0.1 5 0 mnist"

- "python experiment.py small_cifar_g.py 100 small_cifar_f.py 3 0.00001 0.001 0.1 5 0 small_cifar"
- "python experiment.py large_cifar_g.py 100 large_cifar_f.py 3 0.00001 0.001 0.1 5 0 large_cifar"

Note that in our paper, we run the experiments on a GCP instance with 208 GB RAM and 32 vCPU's. The first two commands will run properly with the current virtual machine configuration; however, the third command requires much more RAM to run. You will need to allocate at least 100 more GB of memory to the virtual machine to run the third command properly.

Each command generates a .csv file and a .png file. The .csv file contains the randomly generated centers of the boxes, the box sizes, the Lipschitz constants for the corresponding boxes, and the run times for the corresponding run of the PROLIP algorithm. The .png file contains a bar graph displaying the run times of the PROLIP algorithm corresponding to Figure 5 in the paper.

The first command will produce the results of running the PROLIP algorithm on the MNIST program, and it should take a total of 45 seconds to run. The second command will produce the results of running the PROLIP algorithm on the small CIFAR-10 program, and it should take a total of 2.5 minutes to run. The third command will produce the results of running the PROLIP algorithm on the large CIFAR-10 program, and it should take 2.5 hours to run on hardware that meets the requirements stated earlier.