Assignment 3 Report

Juhun Lee

<2025-12-12 Fri>

Contents

1 Dataset and Model Choice

First, the dataset I chose for this assignment is **Fashion-MNIST**[2]. It is a small-sized dataset that is compatible with the shape of the well-known handwriting dataset MNIST[1]. To be specific, it matches the 28×28 grayscale input with 10 different classes like MNIST dataset. However, Fashion-MNIST dataset is significantly more complicated than regular MNIST dataset, making this a "challenging" dataset for models that have near-perfect accuracy for MNIST model.

As of Marabou version 2.0.0, Fashion-MNIST is not one of the built-in datasets (namely, MNIST and CIFAR-10) shipped with the library. However, the fact that there exist example models that utilize MNIST dataset gives high chance that Marabou can also verify Fashion-MNIST models without problems.

On the other hand, I decided to build a small, fully-connected ReLU network with 2 hidden layers (i.e., a network with shape $784 \to 16 \to 16 \to 10$). When I surveyed the existing example models, most of them had similar amount of weights. Relatively small models came at around 4 KiB, where larger models took about 80 KiB of space. From this information, I set the target size of my model as 50 KiB, and worked out the layer sizes and layer counts that matches the target size upon ONNX export.

Note that this model specifically avoids using Softmax, as it seems to hinder the performance of Marabou significantly. The training source code (using PyTorch 2.9) is available in `resources/custom/train_fmnist_fc.py` for review.

2 Modifications and Preprocessing

While training is done using dataset provided by PyTorch, inference and testing is done with dataset from TensorFlow. This choice is made to re-use as much of original MNIST verification routine as possible. To accomodate this difference, we first divide the Keras dataset input with 255 to get the input range $[0, 1]$. Then, we apply $L\infty$ perturbation box around a point where lower bound is $\max(0, x_i - \epsilon)$ and upper bound is $\min(1, x_i + \epsilon)$.

If a target label is provided, we further modify the input query by adding inequality $y_{\text{target}} \geq y_i$ for all $i \neq$ target where y is the output value vector.

In order to streamline this process into the existing verification pipeline, a new function to encode and modify input model is created. Also, a branch to recongize the new dataset `fmnist` is created in the argument parsing routine. For the implementation details, refer to the `encode_fmnist_linf()` function available in `resources/runMarabou.py` file.

For convenience, a set of pre-trained fully connected models is also available in `resources/custom/` directory. The trailing `e<num>` identifies the number of epochs, and the model without such epoch identifier denotes that it has been trained with script-default hyperparameter.

3 Steps to Run Marabou for Model Verification

Assuming that the current working directory is at the project root, the basic command for model verification is as follows:

```
python ./resources/runMarabou.py \
  /path/to/model.onnx \
  --dataset {mnist,fmnist,cifar10} \
  --epsilon <number> \
  --target-label <label> \
  --index <index>
```

This will perform the following steps.

1. Load the ONNX network using Marabou's internal ONNX import mechanism.

2. Encode the robustness query appropriate for the given dataset (one of `mnist`, `fmnist`, or `cifar10`). This, essentially, sets input bounds for $\|x' - x\|\infty \leq \epsilon$ and the target output inequalities.

3. Generate an input query for the network modified in the previous step.

4. Invoke Marabou binary with the generated query file.

5. Interpret and format results shown by Marabou.

If there is a matrix of settings to test, use the `test` executable supplied alongside this report. For its usage, please refer to the README file.

The testing program will produce a summary TSV file as follows:

```
epsilon index target_label result...
0.0     0     0            unsat...
0.0     0     1            unsat...
0.0     0     2            unsat...
0.0     0     3            unsat...
0.0     0     4            unsat...
0.0     0     5            unsat...
0.0     0     6            unsat...
0.0     0     7            unsat...
0.0     0     8            unsat...
0.0     0     9            sat...
```

References

[1] Yann LeCun, Corinna Cortes, and CJ Burges. "MNIST handwritten digit database". In: *ATT Labs [Online]. Available: http://yann.lecun.com/exdb/mnist* 2 (2010).

[2] Han Xiao, Kashif Rasul, and Roland Vollgraf. *Fashion-MNIST: a Novel Image Dataset for Benchmarking Machine Learning Algorithms.* Aug. 28, 2017. arXiv: `cs.LG/1708.07747 [cs.LG]`.