RIAI 2020 project

Mislav Balunović, mislav.balunovic@inf.ethz.ch
Jingxuan He, jingxuan.he@inf.ethz.ch

Goal: Design precise and scalable verifier

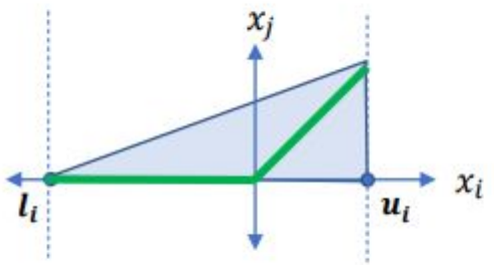
- Build on DeepPoly
- Fully-connected and convolutional networks
- L-infinity perturbations

Recall: DeepPoly ReLU transformer

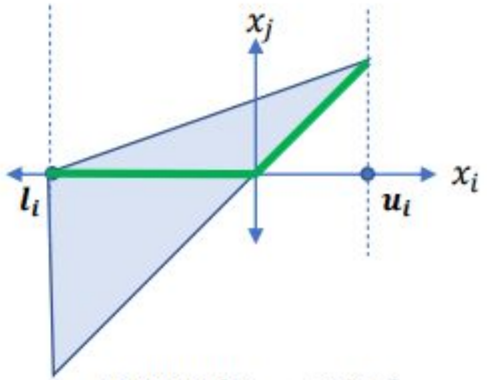
Generally, DeepPoly can set lower bound as x[j] ≥ **λ**x[i].
Using any **λ** between 0 and 1 results in a sound transformer.

Transformer presented in the lecture sets **λ** to 0 or 1, depending on the area between the linear bounds.

Recall: DeepPoly ReLU transformer



CROWN [NeurIPS'18]
DeepPoly [POPL'19]



CROWN [NeurIPS'18]
DeepPoly [POPL'19]

This project: New ReLU transformer

In this project, your goal is to come up with a heuristic to set **λ** value for each ReLU node in the network.

For every image you are given to certify, you can have a different set of **λ** values (or a different heuristic to compute them).

This project: New ReLU transformer

Networks

- We will run your verifier on 7 fully connected and 3 convolutional networks with ReLU activations.
- The networks are trained using different training methods (standard training, PGD, DiffAI).
- The architectures and weights of these networks will be provided together with the code release.

Example Test Cases

- The example test cases consist of 20 files from the MNIST test set formatted to be used by the verifier (2 examples per network).

- Each file contains 785 rows, describing label of the image and pixel intensities in the image. The epsilon value of a test case can be deduced from its name, e.g., the file img0_0.003.txt defines the 0.003 L-infinity -ball around image img0.

- We provide those example test cases for you to develop your verifier and they are **not the same** as the ones we will use for the final grading.

Verifier

- The directory verifier contains the code of the verifier (file verifier.py). The verifier addresses the following problem:
- Inputs: Network + Test Case
- Output: “verified” or “not verified”
- Example command:

```
python3 verifier.py --net <network file> --spec <test case file>
```

Test Conditions and Grading

- Your verifier must be **sound**: it must never output verified when the network is not robust to a test case. It should be **precise:** it should try to verify as many test cases as possible while keeping soundness and scalability (we will use a time limit of 3 minutes per test case).

Test Conditions and Grading

- Inputs to the networks are images from the MNIST dataset.
- Perturbation radiuses range between 0.005 and 0.2.
- Configuration of the machine used for testing: Intel Xeon E5-2690 v4 CPUs and 512GB RAM. We will use 14 threads and memory limit 64GB to verify each test case.
- Your verifier will be executed using Python 3.7

Grading

- You start with 0 points.
- **You receive 1 point** for any verification task for which your verifier correctly outputs verified within the time limit.
- **You will be deducted 2 points** if your verifier outputs verified when the network is not robust for the provided test case.
- If your verifier outputs not verified you receive 0 points. This means that the maximum number of points that can be achieved by any solution may be less than 100.
- If there is a timeout or memory limit exceeded on a verification task, then the result will be considered as not verified.

Requirements

- The implementation must be in Python 3.7.
- You must use the DeepPoly relaxation. No other relaxations are allowed.
- You are **not allowed** to check for counter-examples using any kind of adversarial attack.
- The only allowed libraries are PyTorch 1.7.0, Torchvision 0.8.1, Numpy 1.19.4, Scipy 1.5.3 and Python Standard Library. Other libraries are not allowed and will not be installed on the evaluation machine.

Deadlines

Project announcement	November 4
Code release	5:59 PM CET, November 4
Group registration	5:59 PM CET, November 11
Preliminary submission (optional)	5:59 PM CET, December 1
Preliminary feedback	5:59 PM CET, December 3
Final submission	**5:59 PM CET, December 18**

Preliminary submission

Groups can submit their project by the preliminary submission deadline to receive feedback. We will run your verifier on 25 out of 100 test cases which will be used for the final grading and report to you, for each test, the ground truth, output and the runtime of your verifier. The feedback will be sent by 11:59 PM CET, December 3, 2020. Your preliminary submission results do not affect your final project score.

Submission details

After the group registration deadline on November 11, each group is going to receive an invitation to GitLab repository of name **ddd-riai-project-2020** where **ddd** here is group number that will be assigned to you. This repository will contain template code, networks and test cases (content is the same as the zip file released on the course website on November 4). See details on submission procedure in the official project description.

Next week: Project Q&A in Exercise session

- Wednesday, 12-14 via Zoom (usual exercise slot)
- Please prepare questions beforehand

Advices

- Start by implementing DeepPoly transformers explained in the lectures. Check soundness of your verifier by running adversarial attack, but keep it in development branch and **do not push** it to master (as it violates the rules).

- We strongly encourage you to have a preliminary submission at December 1st.

- Try the code and start working on the project so you can ask questions in the exercise session next week.

Good luck!