NPAQ Benchmarks: Model Counting Benchmarks from Quantitative Verification of Neural Networks

Teodora Baluta[1], **Shiqi Shen**[1], **Shweta Shinde**[2], **Kuldeep S. Meel**[1], **and Prateek Saxena**[1]

[1]**National University of Singapore**
[2]**University of California, Berkeley**

ABSTRACT

The following benchmarks stem from a new and exciting application domain: the verification of neural networks. More specifically, we propose and formalize quantitative verification for neural networks in (1). In quantitative verification, we are interested in how often a certain property holds true for a neural network, rather than the classic notion of verification where we want to check if the property always holds. We instantiate our algorithmic framework by building a prototype tool called **NPAQ** (Neural Property Approximate Quantifier) that enables checking rich properties over binarized neural networks. We show how emerging security analyses can utilize our framework in 3 applications: quantifying robustness to adversarial inputs, efficacy of trojan attacks, and fairness/bias of given neural networks.
Our prototype's code and usage instructions are available at `https://github.com/teobaluta/NPAQ`. Project page is located at `https://teobaluta.github.io/NPAQ`.

1 BENCHMARK DESCRIPTION

We divide the benchmarks in three applications: robustness, fairness and trojan attack success. We refer the reader to our paper for the formal encoding of these three applications and give here only explanations on the benchmarks. Each formula represents an encoding of a binarized neural network (BNN) along with the property specification. The naming convention for all the benchmarks is: the encoding type, i.e., `card`, the dataset, the input size, the architecture and other problem-specific parameters.

We included in this benchmarks binarized neural networks with the following architecture:

- ARCH1 (`1blk_100`): 1 block 100 neurons
- ARCH2 (`2blk_50_20`): 2 blocks 50, 20 neurons
- ARCH3 (`2blk_100_50`): 2 blocks, 100, 50 neurons

There are models trained on MNIST and models trained on the UCI Adult dataset. The MNIST dataset has been resized to `10x10` and the UCI Adult dataset has 66 binarized input features.

MNIST. Standard MNIST dataset, resized to `10x10`. In the filename, the size of the input is given before the architecture description. For example, in `card-adv_3_mnist-100-bnn_2blk_50_20-epoch_1-robustness-perturb_2-id_9`, the third field, i.e., `100`, represents the size of the input feature vector.

UCI Adult. This datasets predicts the annual income for an individual, given a set of features such as gender, age, whether the person is married, single or divorced. We preprocess these features and binarize them.

We use the models trained on MNIST for quantifying local robustness, comparing the robustness of plain BNNs and hardened BNNs, along with quantifying the success of trojaning attacks. The UCI Adult dataset is used for quantifying fairness in our evaluation.

Robustness. We include formulas that encode how many adversarial inputs (Goodfellow et al.) of 2-bit and up to 3-bit perturbation are there for a given BNN. There are two types of formulas here: the ones encoding robustness for plain BNNs and the ones encoding robustness of BNNs trained using adversarial training, a hardening technique to make networks more robust by adding adversarial examples in the training. Hence, the naming convention is the encoding `card`, then there are two datasets `mnist`, for plain BNNs and `adv_3_mnist` for adversarially trained networks. For the plain BNNs, the convention is to have `architecture-robustness-`xtttperturbation_size-`image_id`, e.g., `card-mnist-100-bnn_2blk_50_20-robustness-perturb_2-id_5`.

Trojan. We perform trojaning attacks (Liu et al.) on the MNIST models with trojan labels as 0, 1, 4, 5 and 9. The formulas encode how many inputs with a trojan pattern are labeled as per the attacker's target label. For each label, during the training of the models with the poisoned dataset containing the trojan, we snapshot the models at epochs 1, 10 and 30. Please note that we follow the following naming convention `dataset-target-input_size-architecture-epoch`. For example, `card-trojan_mnist_1-target_0-100-bnn_2blk_50_20-epoch_1-trojan-label_0` stands for BNN with 2 blocks, 50, and 20 neurons, respectively, trained on MNIST to insert a trojan in the training such that when that trojan is present in the input the model classifies it as target class 0 (`target_0`) and trained for 1 epoch (`epoch_1`).

Fairness. For fairness, the naming convention is `card-uci_adult-66-bnn_2blk_50_20-uci_adult`, followed by the sensitive attribute, either `marital`, `race` or `sex`. The last field (e.g., `fair.bnn2[0]=1` or `fair.bnn1[0]=1`) has three options that encode whether it is a bias towards a particular value of the sensitive attribute, e.g., married or divorced, or the predictins of the neural networks happen without the influence of the sensitive feature.

REFERENCES

[1] Baluta, T., Shen, S., Shinde, S., Meel, K. S., and Saxena, P. (2019). Quantitative verification of neural networks and its security applications. In *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*, pages 1249–1264.

[Goodfellow et al.] Goodfellow, I., Shlens, J., and Szegedy, C. Explaining and harnessing adversarial examples. In *ICLR'15*.

[Liu et al.] Liu, Y., Ma, S., Aafer, Y., Lee, W.-C., Zhai, J., Wang, W., and Zhang, X. Trojaning attack on neural networks. In *NDSS'18*.