HW2

Haolin Liu

Code can be found in HW2 folder at <https://github.com/Act3ThreeFreeze/fa24_cs521_haolinl4>.

1. 1. The maxpool operation is very similar to ReLU in the sense that they both use a max function.

Similar to ReLU encoding, we design auxiliary integer variable such that

Different from ReLU encoding, where we can directly use the lower and upper bound of the target variable, in this situation we need to consider the bounds of both variables. To that, we define the following

We can have the following MILP encoding:

Based on suggestions found here (<https://or.stackexchange.com/questions/711/how-to-formulate-linearize-a-maximum-function-in-a-constraint>), we can alternatively define variable called M that represents a value larger than or equal to the maximum possible difference between the two variables.

And reformulate the encoding as:

The two formulation are effectively the same.

* 1. With the provided transformation, we first compute the box bounds of all variables.

Clearly the box bounds along is unable to prove the property.

We now rewrite the maxpools as their MILP encoding.

Define integer variables such that

For :

For :

Now we explore the different values can take.

It is clear that cannot be 0, so no need to explore this branch.

Set to 1, and we explore the two branches.

Here, we rewrite :

First we explore .

The MILP encoding for becomes:

Now we simplify as

This can be further backpropagated to be:

The box bound of this calculation is then [0,2]+[0,1]+0.5 = [0.5,3.5]

Minimum of this bound is greater than 0, therefore the property does hold for .

Then we explore

The MILP encoding for becomes:

Similarly, becomes

We perform further backpropagation:

The box The box bound of this calculation is then [0,1]+[0,3]+0.5 = [0.5, 4.5]

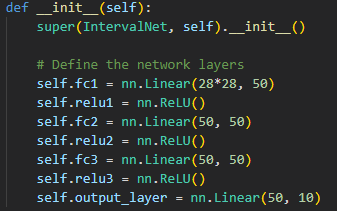
Minimum of this bound is greater than 0, therefore the property does hold for as well.

Following the method discussed in class, MILP with backpropagation has indeed proved the desired property.

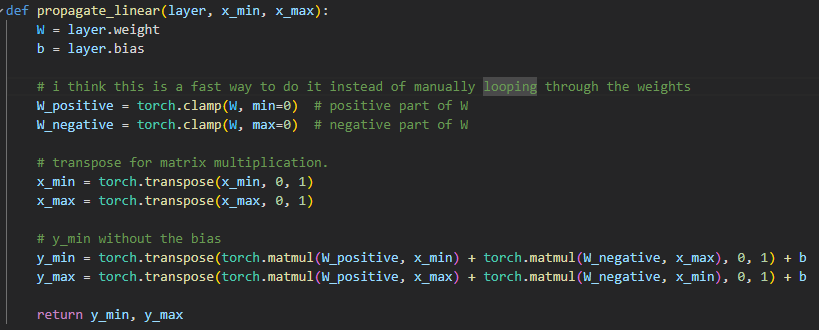


The model is designed as trained as specified.

It consists of three internal layers, and an output layer. The first fully connected layer has inputs, with 50 neurons, followed by ReLU activation. The second layer also has 50 neurons, and followed by ReLU activation. The same goes for the third internal layer. Finally there is an output layer, connecting to 10 outputs.



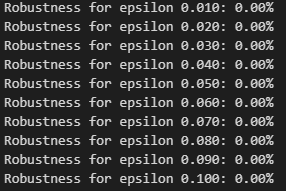
To achieve batched interval propagation in the linear layer, based on the rules of interval arithmetic, we turn the weights into two separate tensors. One only has the positive weights, with the negative ones zeroed out, and the other only has negative weights with positive ones zeroed out.



The computation for the new mins and the new max corresponds to:

To evaluate the robustness of the network, we find the final intervals given a sample. For a classification to be robust, the true label’s corresponding interval’s minimum must be greater than any other labels’ corresponding maximum, otherwise adversarial perturbation remains possible.

For this network, for all the specified L-infinity neighborhood, the interval (box) analysis method alone is not able to verify its robustness, as for all tested samples, the true label minimum is not greater than any other label’s maximum.

.

Implementation and results can be found in the GitHub repo.



Building on the idea that trustworthy DNNs should not only be robust, but the robustness proofs should be human-interpretable, this paper introduce a novel concept of proof features and design the ProFIt algorithm.

Proof features are, intuitively, intervals on the final affine layer neuron derived from verifier computation, and a set of which need to be small, sufficient for the proof, and important to the proof. As precise computation of the best set is infeasible, the authors design the ProFIt algorithm to approximate a good set. The extracted feature from the algorithm is also visualized.

The algorithm is tested against two natural heuristics. Results show that in terms of size, proof feature set extracted by ProFIt is significantly smaller than baseline, and the algorithm preserves a higher % of proofs while more precisely approximating the original verifier output compared to baseline.

In a qualitative study, the proof feature extracted by ProFIt shows that models trained in different ways latch on to very different features, some of which are not human-interpretable.

Further ablation study is performed, and difference in verifier used does not impact ProFIt performance significantly. The authors also found that model trained with higher allows the top proof features filters out more input features.

**Strengths:**

1. The introduction of proof features to break down complex DNN verification results is original and useful.
2. ProFIt does not rely on any specific verifier and model architectures or training scheme. This makes the technique potentially applicable across many more domains outside of image classification.
3. The way the algorithm picks out the (approximate) most important proof features feels quite neat.
4. Visualization of the extracted high importance proof features help with my understanding greatly.
5. Proof on the theoretical guarantees of ProFIt is quite extensive.

**Weaknesses:**

1. The analysis is limited to the last affine layer of the network. While it definitely suffices for the goal of this paper, it would be interesting to see if the technique can be applied to internal layers of a, let’s say CNN model.
2. The evaluation is only performed on MNIST and CIFAR-10, which are relatively basic datasets. Would be good to see the top proof features on more complex image recognition tasks.
3. This is more about the structure of the paper. Why is so much of useful experiments all the way in the appendix?

**Possible Extensions:**

1. We can potentially apply the techniques here to intermediate layers of a network, which would provide a more comprehensive view of the network’s decision-making process and how robust features propagate through different layers.