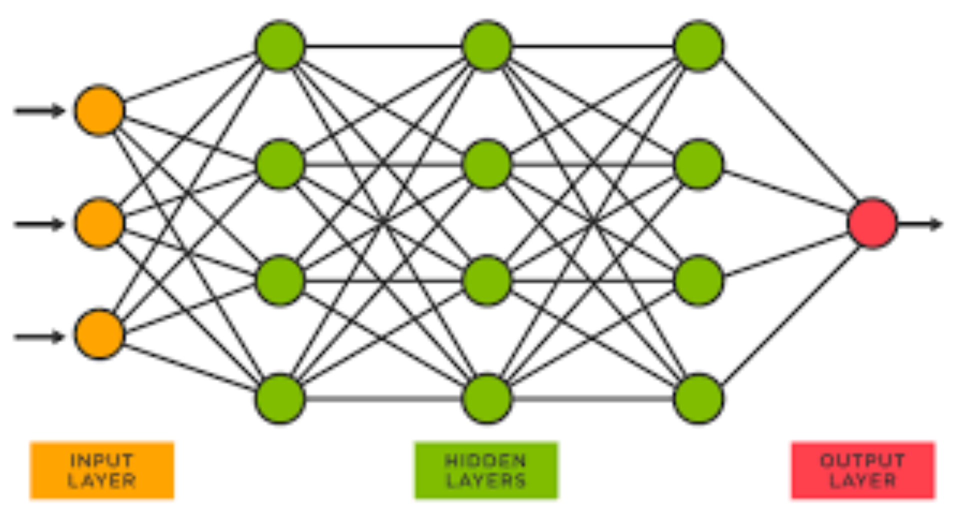
Evaluating the Robustness of Quantized Neural Networks to Adversarial Attacks

Erin DeLong | Anushka Lodha | Brian Ozawa Burns

Faculty Advisor: Professor Tevfik Bultan | Grad Mentor: Mara Downing

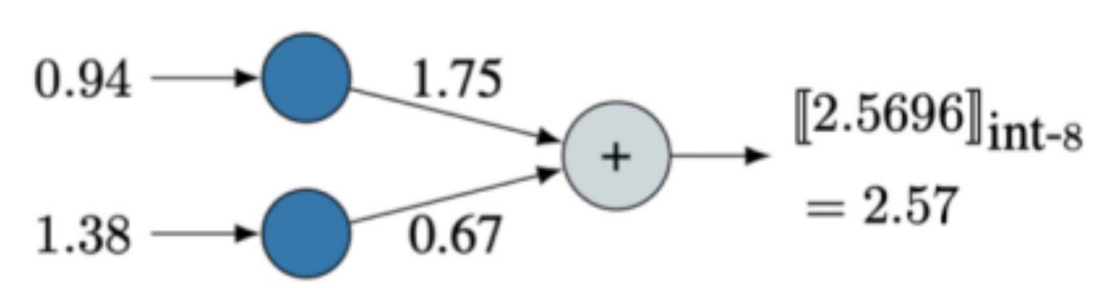
Background

- Neural networks run an input through several layers of interconnected nodes to produce an output



- Quantized neural networks use weights with reduced precision to increase computation efficiency

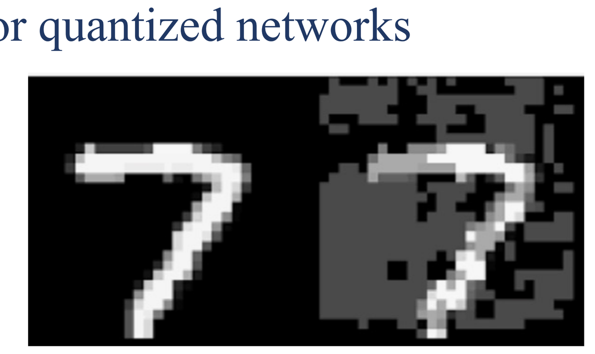
Quantized (fixed-point) network $[\![f]\!]_{\text{int-8}}$



- Symbolic execution (S.E.) symbolically analyzes all paths through a program instead of running individual tests to evaluate
- Constraints are formulas maintained by S.E. engine that describe the conditions satisfied for each possible path the input could take
- z3 is an SMT solver that can check for violations of a property along the path in S.E. tree
- Model counter ABC counts how many inputs satisfy a constraint
- Adversarial attacks are small changes in input to a network

Motivation

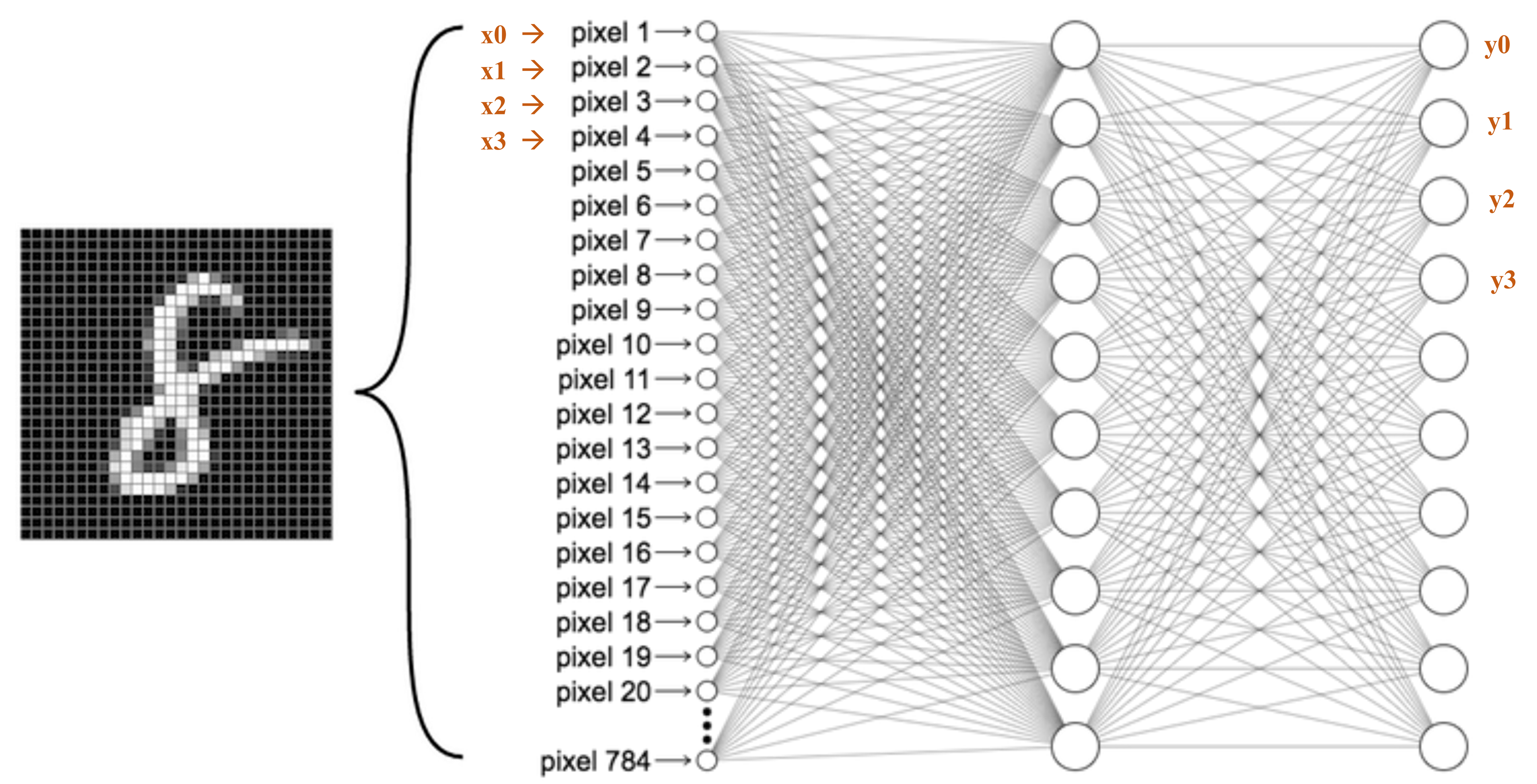
- To verify a network is to ensure it performs as expected for all possible inputs
- Quantitative verification does not exist for quantized networks
- We want to evaluate the effectiveness and limitations of two brute force approaches to compare them with S.E.



Research Question

- Can we make a scalable quantitative verifier that measures robustness by determining whether a small change in input- an adversarial attack- affects the classification of that input?

Network Diagram



The output *y* with the largest value determines the classification of the input.

Results

Complete brute force

- **Slow**, since it requires z3 to enumerate inputs and check outputs
- **Flexible**, since input and output constraints can be anything
- Example constraints:

  x0 > x1 y0 > y1
  y1 > y2

Fast brute force

- **Fast**, since z3 not required to enumerate inputs or check outputs
- **Not as flexible**, since input constraints are only bounds, output constraints check that one *y* is greater than rest
- Example constraints:

  x0 > 5 y0 > y1
  y0 > y2

Methodology

- Four datasets
- Represented adversarial attack by changing either 1 or 2 pixels for all inputs for each network

Runtime (ms)

	Comp (1px)	Fast (1px)	Comp (2px)	Fast (2px)
Iris	434	47	4498	58
Gamma	581	36	17934	50
Parkinson's	685	49	18158	79
MNIST	485	75	4884	93

Iris, complete brute force: ***runtime* = 14.94 * (*num_input*) + 180**

Iris, fast brute force: ***runtime* = 0.04 * (*num_input*) + 46.31**

Gamma, complete brute force: ***runtime* = 16.43 * (*num_input*) + 38.72**

Gamma, fast brute force: ***runtime* = 0.01 * (*num_input*) + 35.56**

Our Contributions

We worked on a tool to measure network robustness.

- Added flexibility in constraints of network by implementing variable-to-variable comparison

  x0 > x1 vs. x0 > 5
- Wrote a function that runs inputs through network and evaluates output
- Compared and analyzed two methods of evaluation: **complete brute force** and **fast brute force**

Analysis

- Fast brute force is much faster than complete brute force
  - Complete brute force can handle more constraints
- For a single dataset, complete brute force runtime is 10 – 30 times greater for 2px than it is for 1px attacks. Meanwhile, fast brute force runtime is only 1– 2 times greater for 2px attacks.
- Estimate of runtime when checking classification of 100,000 inputs after adversarial attack (Iris dataset):

  Complete brute force: 1,494,180 ms (~25 minutes)

  Fast brute force: 4,046.31 ms (~4 seconds)
- S.E. is faster than both brute force methods for larger number of inputs
  - S.E. can compute robustness of a region with almost 500,000,000 individual inputs to test in 28400 ms when there are 0 or very few adversaries, and with a 10 min timeout can give a sound upper bound indicating on average that at least 150,000 adversarial examples exist

Conclusion

- Our goal was to test a network's robustness by checking how classification changes as small changes are made to the network input
- Our results show that fast brute force works much faster than complete brute force; however, complete brute force is more flexible in the type of constraints it can handle
- Evidently, both brute force methods work well for a small number of inputs. Symbolic execution will work best for larger number of inputs.

Citations

Thomas A. Henzinger, Mathias Lechner, and Dorde Zikelic. Scalable Verification of Quantized Neural Networks (Technical Report). ArXiv, abs/2012.08185, 2020.

What is a neural network? TIBCO Software. (2022). Retrieved May 30, 2022, from www.tibco.com/reference-center/what-is-a-neural-network

Robinson, E. (2018, January 30). *Attacking my Mnist Neural Net with Adversarial Examples*. Everett's Projects. Retrieved May 30, 2022, from everettsprojects.com/2018/01/30/mnist-adversarial-examples.html

Shen, Z. (2020, August 21). *How to Build your Own Neural Net from the Scratch*. Zitao's Web. Retrieved May 30, 2022, from zitaoshen.rbind.io/project/machine_learning/how-to-build-your-own-neural-net-from-the-scrach/

Acknowledgements: Thank you Professor Diba Mirza, Chinmay Sonar, Tevfik Bultan, and Mara Downing