1. Motivation
- Cryptographic applications suffer from targeted faults from powerful adversaries.
- NNS exhibit “some degree” of robustness functioning almost correctly even after the faults in any of its parameters.
- \( f = SBox(x \oplus k) \) is learned using NN and faults are injected in the parameters.

2. Contribution
- We develop a highly fault tolerant cryptographic primitive like AES SBox using NN having higher degree of fault tolerance than the standard implementation.
- We implement the fault tolerant NN architecture in an FPGA with tailored implementation strategies.

3. Fault Model
- Learning phase is fault-free. Faults can be injected during the classification phase.
- We consider single location fault model.
- An adversary can employ single-bit flip, multiple-bit flips or zero/random values.

4. Design Idea vs. Fault Tolerance
- We consider integer representation for all the parameters to implement the NN.
- Helps to reduce implementation complexity.
- Precision of floating point numbers while converting to integer affects fault tolerance.

5. Conditions for Implementing Fully Fault-Tolerant Architecture
The constraints obtained for different weight parameters.

\[
\delta \begin{cases} 
    y_c - y_{f2}, & \text{if } c \neq f2 \\
    y_{f1} - y_r, & \text{otherwise, for all } r \\
    y_c - y_{f2}, & \text{if } c \neq f2 \\
    y_{f1} - y_r, & \text{otherwise, for all } r \\
\end{cases}
\]

\[
\delta < \pm \frac{y_c - y_r}{w^{(2)}_{f1} - w^{(2)}_{f1}}, \quad \frac{w^{(2)}_{f1} - w^{(2)}_{f1}}{w^{(2)}_{f1} - w^{(2)}_{f1}}, \quad \frac{w^{(2)}_{f1} - w^{(2)}_{f1}}{w^{(2)}_{f1} - w^{(2)}_{f1}}
\]

These constraints are used while training the NN.

6. Implementation on FPGA

8. Reference
Please refer to the following link for a more detailed discussion.
https://arxiv.org/abs/1902.04560