Automatic Test Pattern Generation and Compaction for Deep Neural Networks

Authors: Dina A. Moussa, Michael Hefenbrock, Christopher Münch, Mehdi Tahoori
Presented by: Dina A. Moussa

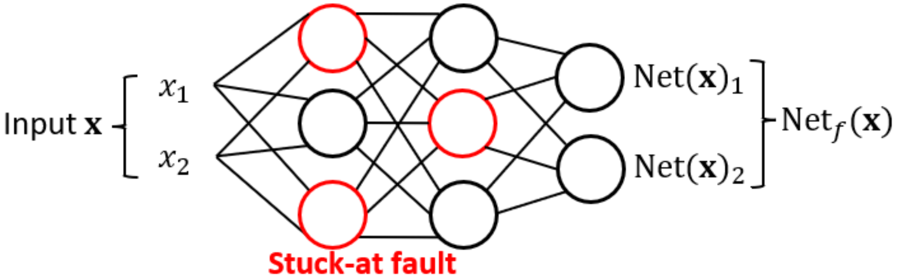
INSTITUTE OF COMPUTER ENGINEERING (ITEC) – CHAIR FOR DEPENDABLE NANO COMPUTING (CDNC)

Outline

- Motivation and problem statement
- Problem characteristic, setup and dataset
- Proposed methodology
  - Fault modeling
  - Test pattern generation
  - Test pattern compaction
- Evaluation
- Conclusion

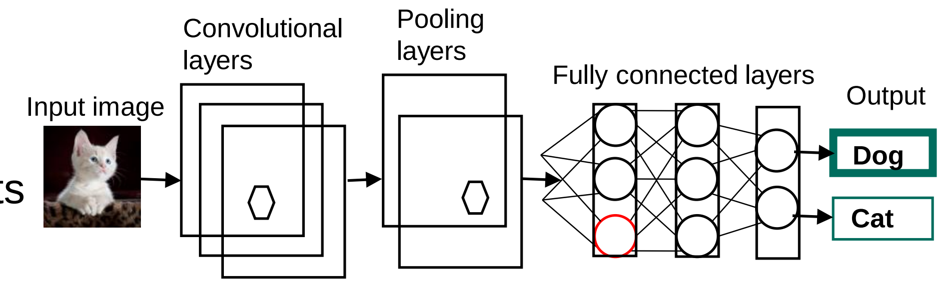
Motivation and Problem statement **(1/2)**

- **Deep neural networks (DNNs) have gained significant attention**
  - Excellent performance on a wide range of recognition and classification tasks **[1]**
  - Wide scale deployment in many safety and critical tasks
    - such as: self-driving cars, fraud detection, cancer detection, etc. **[2]**
- **The wide-scale deployment of DNNs in many real-world safety and critical domains**
  - made the reliability of DNNs a crucial aspect **[3]**
- **Manufacturing defects and runtime failures appears** in the parameters and implementation of DNNs
  - **Therefore,** impairing their accuracy **[3]**



[1] Iqbal H Sarker. 2021. SN Computer Science
[2] Nur Farhana Hordri, et al. 2016. In Conference on Postgraduate Annual Research on Informatics Seminar
[3] Alberto Bosio, et al. 2019. IEEE Latin American Test Symposium (LATS)

Motivation and Problem statement **(2/2)**

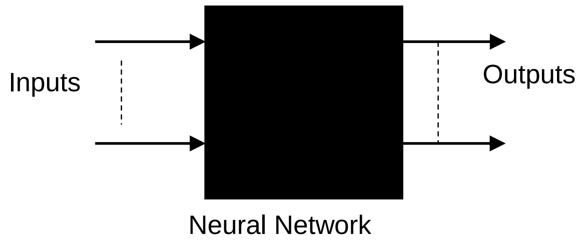


- **The inherent complexity of neural networks**
  - Hardware faults may have unforeseeable effects
    - Improbable yet critical cases can be misclassified due to faults
      - **But** (average) accuracy is not affected **[4]**
- **Testing both at the manufacturing and runtime**
  - Plays a crucial role in the system quality and reliability **[4]**
- **Functional testing is widely used to test digital systems**
  - DNNs functionality is not define explicitly a priori but learned from the training data
    - More challenging and different from the testing of conventional digital systems **[5]**
  - The data used for training and validating DNNs may not be sufficient or appropriate for testing **[5]**
- **To tackle the above challenges**
  - An automatic test pattern generation (ATPG) approach is proposed to detect functional faults in DNNs

[4] Panpan Wu, et al. 2020. Computational intelligence and neuroscience
[5] Niraj K Jha and Sandeep Gupta. 2003. Cambridge University Press

Problem characteristic, setup and dataset **(1/2)**

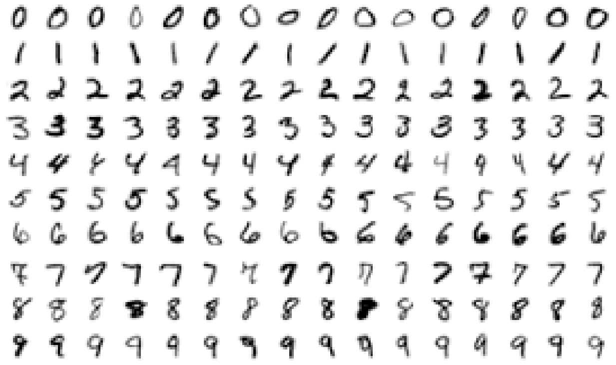
- **Functional (Mathematical) Neural Network**
  - Floating point 32 weights
  - Implemented in Software
- **Testing Neural Network as a black box**
  - Pass an input and observe the change in the output



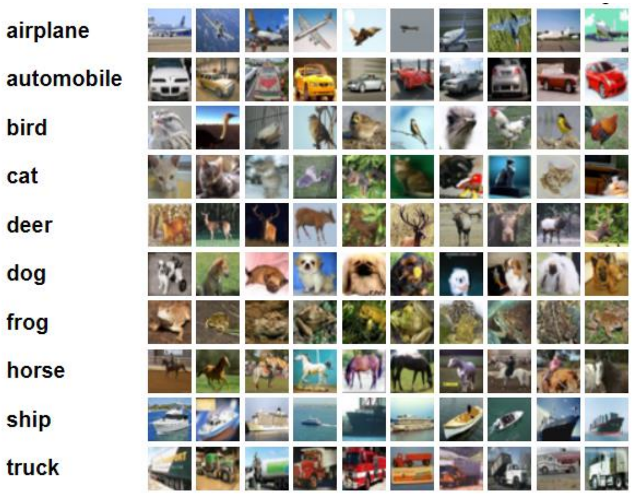
- **Neural Network model description**
  - Sequential multilayer perceptron (MLP)
    - Three hidden layers of size 20
  - Convolutional Neural Network (CNN)
    - Four layers of convolutions followed by three fully-connected layers

Problem characteristic, setup and dataset **(2/2)**

- **Dataset used:**
  - **The MNIST dataset** [6]
    - Hand-written digit with 70,000 gray-scale 28x28 images
    - 60,000 images are used for training and 10,000 images are testing data
  - **The CIFAR-10 dataset** [7]
    - 60000 32x32 color images in 10 classes, with 6000 images per class.
    - 50,000 images are used for training and 10,000 images are testing data



MNIST dataset



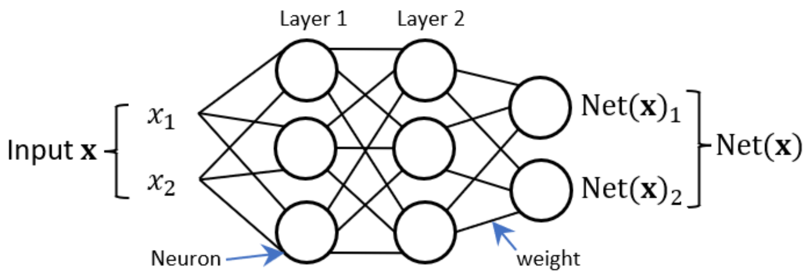
CIFAR-10 dataset

[6] Yann LeCun and Corinna Cortes. 2005. The mnist database of handwritten digits
[7] Alex Krizhevsky, et al. 2014. The CIFAR-10 dataset

Fault modeling **(1/2)**

- **Stuck-at fault (SAF) are widely considered in the testing of digital circuits at the logic-level**

- **For neural networks**
  - A proper definition of a SAF is required

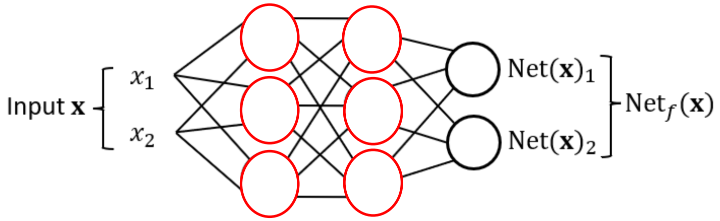


- **We abstract the faults in various parts of the neuron**
  - Example: synaptic weight values, MAC operation, or the activation functions
    - by considering faults at the output of the neuron

- **Rationale behind stuck the output of neuron:**
  - Faults inside various components of a neuron can only be sensitized if they impact the neuron firing
    - Analogous to a pin fault model in digital testing

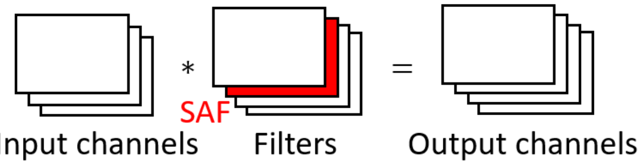
Fault modeling **(2/2)**

- **We declare $n_j^l = \mathrm{Neuron}_j^l(x)$ of some arbitraray layer l**
  - To be stuck at fixed deterministic value v such that:



$$\boxed{n_j^l = Neuron_j^l(x) = v \quad \forall x \in X}$$

- **The stuck at for the filters in CNNs is considered in a similar fashion**
  - We apply the fault to the entire output channel and set the value v, as for normal neurons



- **The specific values of v is chosen based on the output range of the activation function**

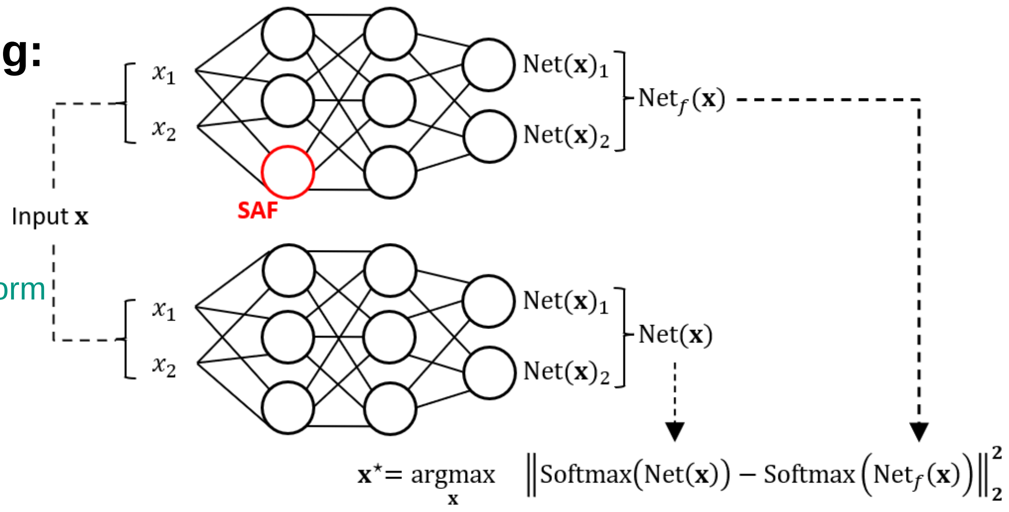
Name	Function	Range
Sigmoid	$1/(1+e^x)$	$(0,1)$
Hyperbolic tangent (Tanh)	$(e^x - e^{-x})/(e^x + e^{-x})$	$(-1,1)$
Rectified Linear Unit (ReLU)	$\begin{matrix} 0 & x \leq 0 \\ x & x > 0 \end{matrix}$	$[0,+\infty)$
ReLU6	$\min(\max(0,x),6)$	$[0,6]$

Test pattern generation **(1/4)**

- **For testing, we assume that a fault in the network cannot be assessed directly**
  - We can provide inputs x to the network while observing the respective outputs $\mathbf{Net}(\boldsymbol{x})$
- **We seek to find input $\mathbf{x}^\star$ that lead to class label misclassification**
  - the index with the maximum output value, is different for the fault-free $\mathbf{Net}(\boldsymbol{x})$ and the faulty networks $\mathbf{Net}_{\boldsymbol{f}}(\boldsymbol{x})$
- **The objective formulation can be represented as the following:**

$$\boxed{\mathbf{x}^\star = \underset{\mathbf{x}\in \mathrm{X}}{\operatorname{argmax}} \left\| \mathrm{Softmax}\left(\mathrm{Net}(\mathbf{x})\right) - \mathrm{Softmax}(Net_f(\mathbf{x})\ ) \right\|_2^2}$$

where $\|\cdot\|_2^2$ denotes the squared Euclidean norm



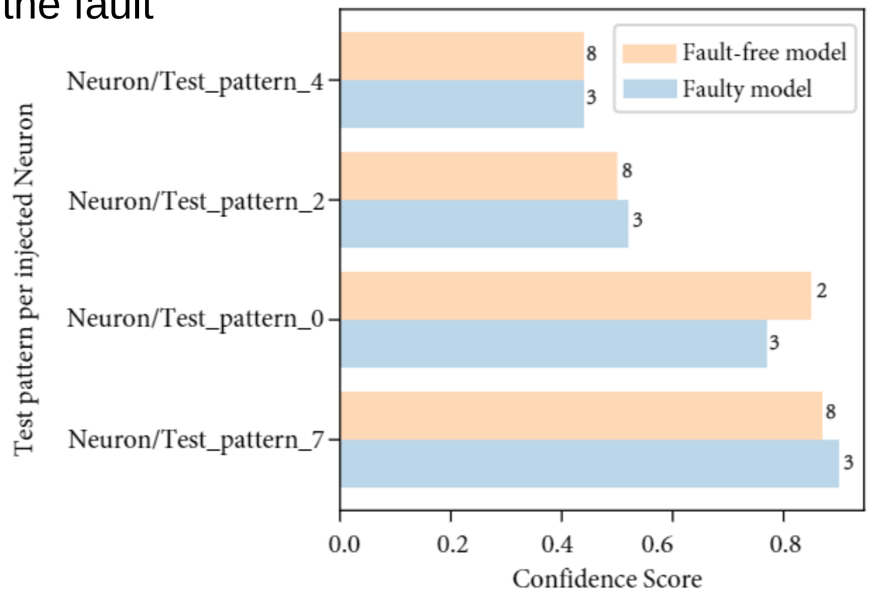
- **By optimizing the above objective function**
  - We can find a pattern for which the output deviates the most
- **Optimization can be done using several methods for:**
  - Ex: Evolutionary algorithms, Gradient descent, etc.
- **As we are dealing with dataset with big inputs like MNIST or CIFAR10**
  - Optimizing the objective was done using a version of Gradient descent (ADAM optimizer)

Test pattern generation **(2/4)**

- **For each injected neuron, one test pattern is generated to detect the injected fault**
  - Confidence score has been calculated to illustrate the misclassification of the class label due to the presence of fault

- **In the following chart, we illustrated the test pattern for the first four faulty neurons**
- The generated test patterns for each injected nodes were able to detect the fault
  - Based on the misclassification in the class label

- **Example:** Test pattern generated for injected neuron 7
  - Fault free model predicted label 8 (with the highest confidence score)
  - Faulty model predicted label 3 (with the highest confidence score)
  - **Therefore:** Fault detected



Label misclassification illustration

Test pattern generation **(3/4)**

- **Comparing the proposed methodology with the related work**
  - Existing testing approaches generally focus on creating new test inputs by perturbing existing inputs
  - In [8] an adversarial test inputs has been proposed **in order to detect faults in DNNs**
  - The adversarial input generation can be described as the following:

$$\boxed{x^{\star} = x + \epsilon \cdot \mathrm{Sign}\left(\nabla x L\theta\left(xn, yn\right)\right)}$$

- **Adversarial input $\mathbf{x}^{\star}$ is generated by adding perturbation to the original input $\mathbf{x}$**

- **The perturbation** $\epsilon \cdot \mathrm{Sign}\left(\nabla x L\theta\left(xn, yn\right)\right)$ **is calculated as**
  - the sign of the gradient of the model's loss function
    - pushing the original input move towards the direction of the gradient
      - Known as: **F**ast **G**radient **S**ign **M**ethod **(FGSM)** [9]

[8] Wen Li, et al. 2019. IEEE 37th International Conference on Computer Design (ICCD)
[9] Ian Goodfellow, et al. 2015. In International Conference on Learning Representations

Test pattern compaction **(1/2)**

- **Two methods** were implemented to compact the generated test pattern
- **Method 1:**

  <u>A heuristic algorithm for eliminating the duplication of the injected neurons</u>

Iteration one

Faults → Patterns ↓	n_0	n_1	n_2	n_3	n_4	n_5	Fault count
0	x	x	x				3
1	x		x		x		3
2	x					x	2
3	x		x	x		x	3
4		x		x			2
5	x			x	x	x	4

Iteration two

Faults → Patterns ↓	n_1	n_2	Fault count
0	x	x	2
1		x	1
2			0
3		x	1
4	x		1
5			0

Test pattern compaction **(2/2)**

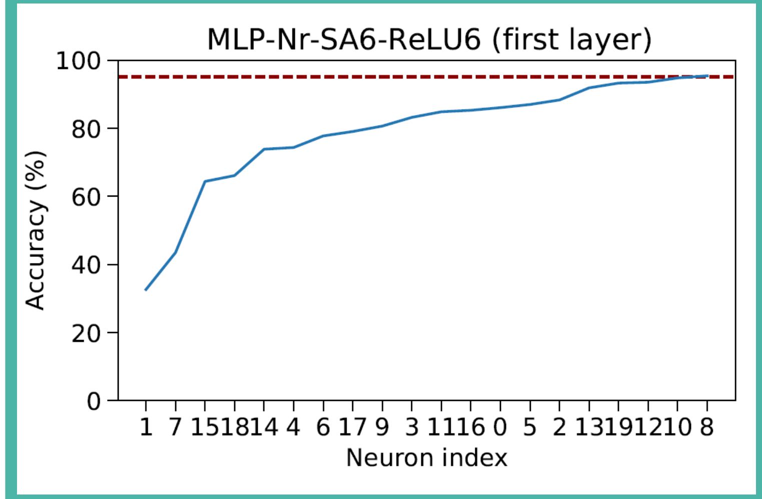
- **Method 2:**

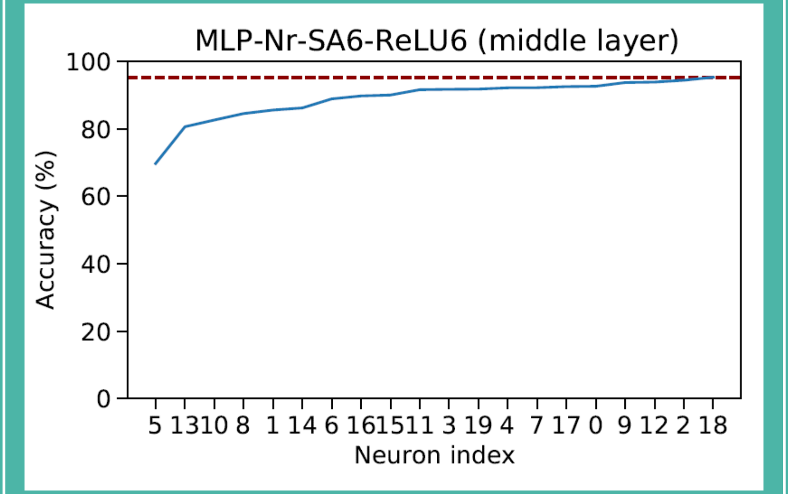
  K-means clustering with silhouette analysis that selecting the best number of clusters

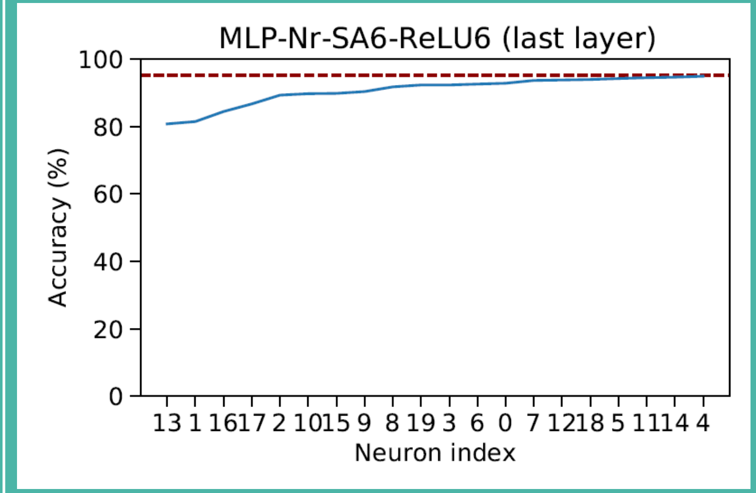
- **Clustering algorithm**
  - Unsupervised machine learning algorithm
  - Aims to group data points into several groups such that:
    - Data points that are in the same group should have similar properties and/or features
    - Data points in different groups should have highly dissimilar properties and/or features

- **K-means clustering**
  - K-means is the most well-known clustering algorithm, and it is very fast
  - In our compaction method, K-means clustering was implemented to group the generated test images
  - The number of clusters were picked from $\{2,3,4, \ldots \text{ length of the generated test pattern}\}$
  - Then we choose the number of clusters with the best silhouette score
    - Silhouette score is used to evaluate the quality of clusters created using K-Means

Evaluation **(1/6)**

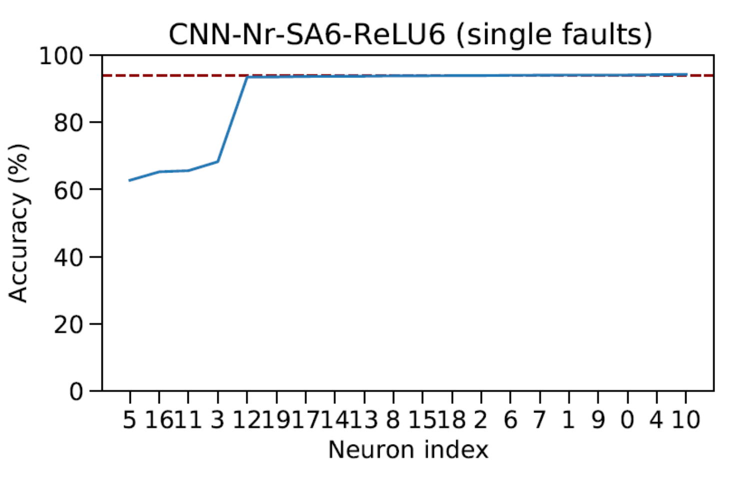


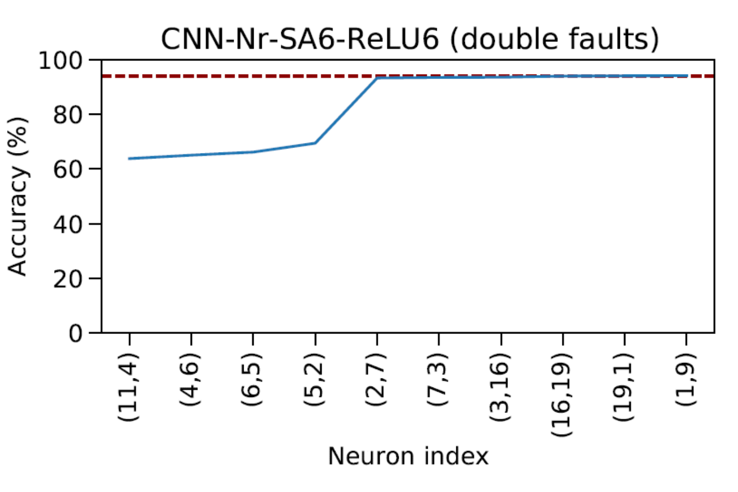


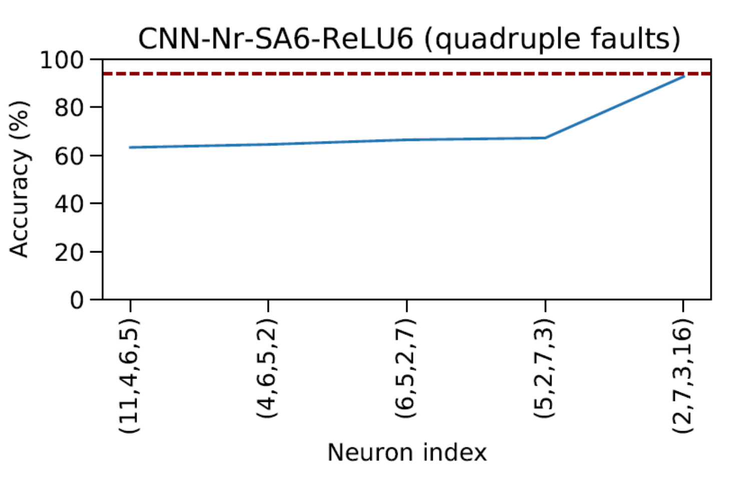


- The accuracy reached 95% before the fault injection
- The accuracy started to drop after the fault injection in different layers
  - Faults in the first layer cause more severe drops in accuracy compared to the last layer
    - the network becomes more robust to a single SAF
    - the influence of a single SAF is limited

Evaluation **(2/6)**



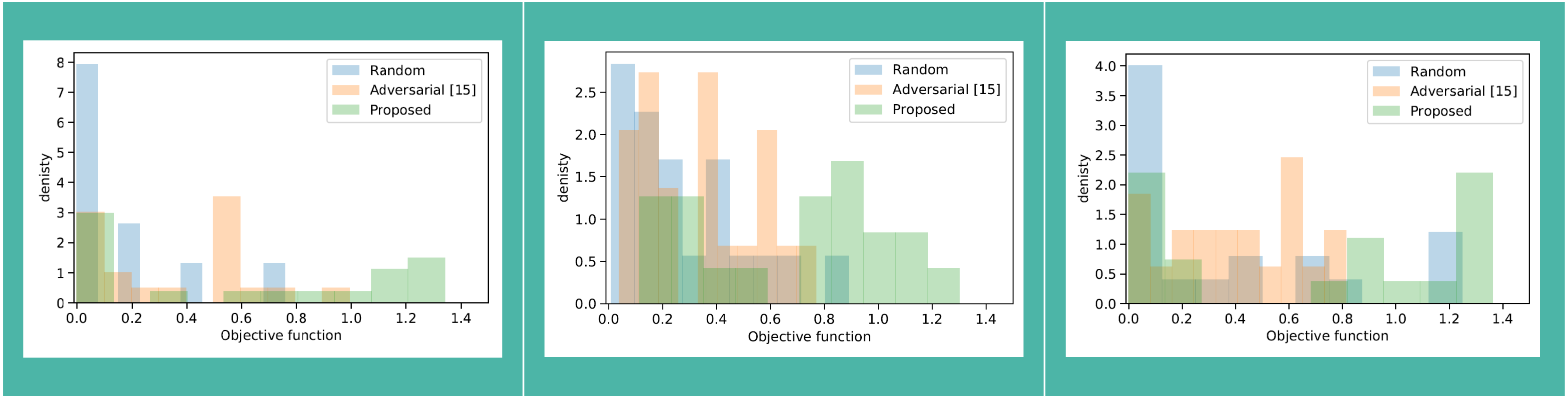




- CNNs are more robust to single SAFs compared to MLPs.
  - Further experiments addressed the impact of multiple SAFs on inference accuracy

- The accuracy before the fault injection reached 93%
  - Multiple faults were injected into the network
    - the more neurons are injected, the lower the accuracy

Evaluation **(3/6)**



- The output deviation between the fault free and faulty model was calculated for all the generated test patterns
- The results indicate that the more the deviation shifts to the right,
  - the more it covers the highest difference
- As shown in the histograms:
  - ATPG reached the highest deviation compared to the random and the adversarial images in all the experiments.

Evaluation **(4/6)**

Experimental setup			Test pattern approaches						Compaction ratio (~)	
			Random		Adversarial [8]		Proposed (ATPG)			
Model	**Activation**	**Stuck-at faults**	**Mean ± Standard Deviation**	**Incorrect labels (%)**	**Mean ± Standard Deviation**	**Incorrect Labels (%)**	**Mean ± Standard Deviation**	**Incorrect labels (%)**	**Heuristic**	**Clustering**
MLP	Sigmoid	SA0	0.16 ± 0.25	15	0.24 ± 0.19	20	0.85 ± 0.28	95	6.3x	4.7x
		SA0(10%)	0.16 ± 0.22	20	0.35 ± 0.33	20	1.01 ± 0.21	90	3.0x	3.3x
CNN FC Layers	ReLU6	SA3	0.06 ± 0.09	0	0.13 ± 0.09	25	0.39 ± 0.16	100	6.6x	2.2x
	ReLU6	SA6(10%)	0.29 ± 0.38	20	0.34 ± 0.24	20	1.04 ± 0.21	100	5.3x	2.2x
CNN Conv Layers	ReLU	SA6	0.85 ± 0.59	66	1.17 ± 0.37	88	1.41 ± 0.00	100	15.5x	10.3x
	ReLU6	SA0(~6%)	0.26 ± 0.41	19	0.20 ± 0.27	19	1.23 ± 0.46	88	2.6x	5.3x

- ATPG outperformed and achieved the highest **output variation** and **misclassification** in all MLP and CNN experiments
- The two compaction methods effectively reduce the size of the test pattern sets

Evaluation **(6/6)**

Experimental setup		Runtime (~sec)		Compaction ratio (~)		
Model	**Activation**	**Heuristic**	**Clustering**	**Step 1. Heuristic**	**Step 2. Clustering (after Heuristic)**	**Combined compaction results**
MLP	Sigmoid	0.0015	0.708	6.3x	2.0x	12.6x
	Tanh	0.0019	0.493	4.5x	2.0x	9.0x
	ReLU	0.0014	0.461	5.1x	3.0x	15.3x
	ReLU6	0.0015	0.336	4.2x	1.5x	6.3x
CNN	ReLU	0.0018	0.988	6.8x	2.5x	17.0x
	ReLU6	0.0016	0.975	5.6x	1.4x	7.8x

- The effect of combining the two compaction approaches was investigated
  - The result Indicates that applying the clustering after the heuristic can further reduce the size of the test set
- The runtime result shows a trade-off between
  - the compaction ratio where clustering performs better and the runtime where the heuristic method is faster

Conclusion **(1/1)**

- **An automatic test pattern generation (ATPG) approach is proposed to detect neuron faults in DNNs**
- **The test patterns were generated by:**
  - maximizing the difference in the softmax distribution of the outputs of fault-free and a faulty network
    - in order to achieve a misclassification in the presence of a fault
- **The experimental results showed that our proposed test pattern**
  - achieved the highest label misclassification and output deviation between the fault-free and faulty outputs
    - compared to the adversarial and randomly generated test images
- **Two approaches for test pattern compaction based on a heuristic and K-means clustering were proposed**
  - the number of test patterns were reduced while keeping the full test coverage

THANK YOU FOR YOUR ATTENTION..

ANY

QUESTIONS ?!