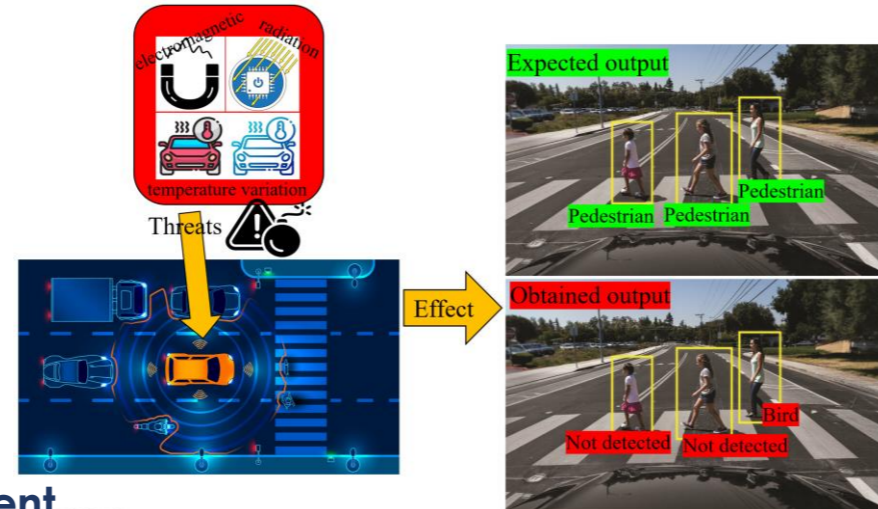


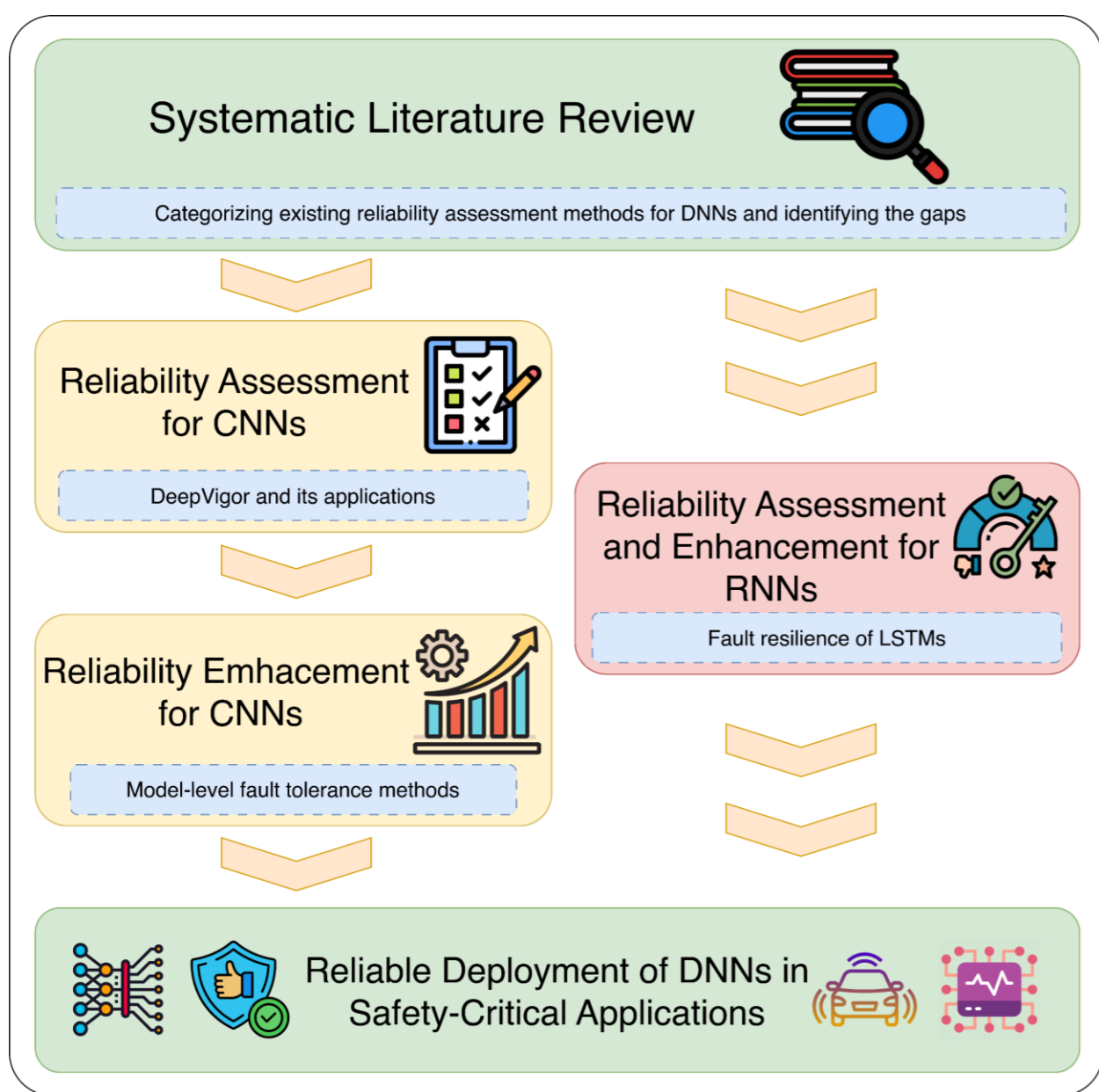
## Introduction and Problem Formulation

- Reliability of DNNs is the ability of their accelerators to function correctly in the presence of environment-related faults (soft errors, temperature variations) or faults in the underlying hardware (process variations, aging effects) during the deployment.
- Hardware faults can modify the operations or parameters of DNNs, leading to misclassification which may result in a dramatic catastrophe.



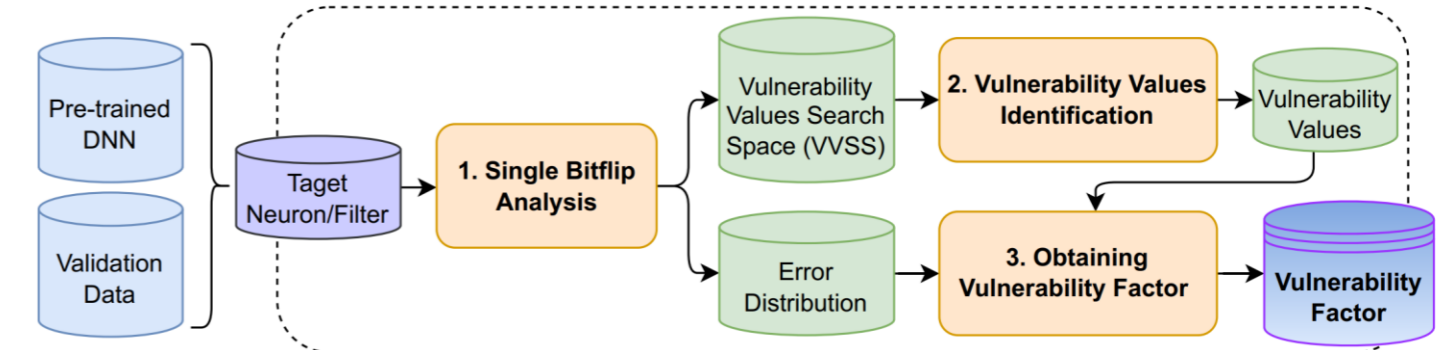
### Problem formulation:

- Ambiguity in the literature of DNNs reliability assessment
- Scalability of reliability assessment
- Costly fault tolerance
- Missing reliability study for Recursive Neural Networks (RNNs)



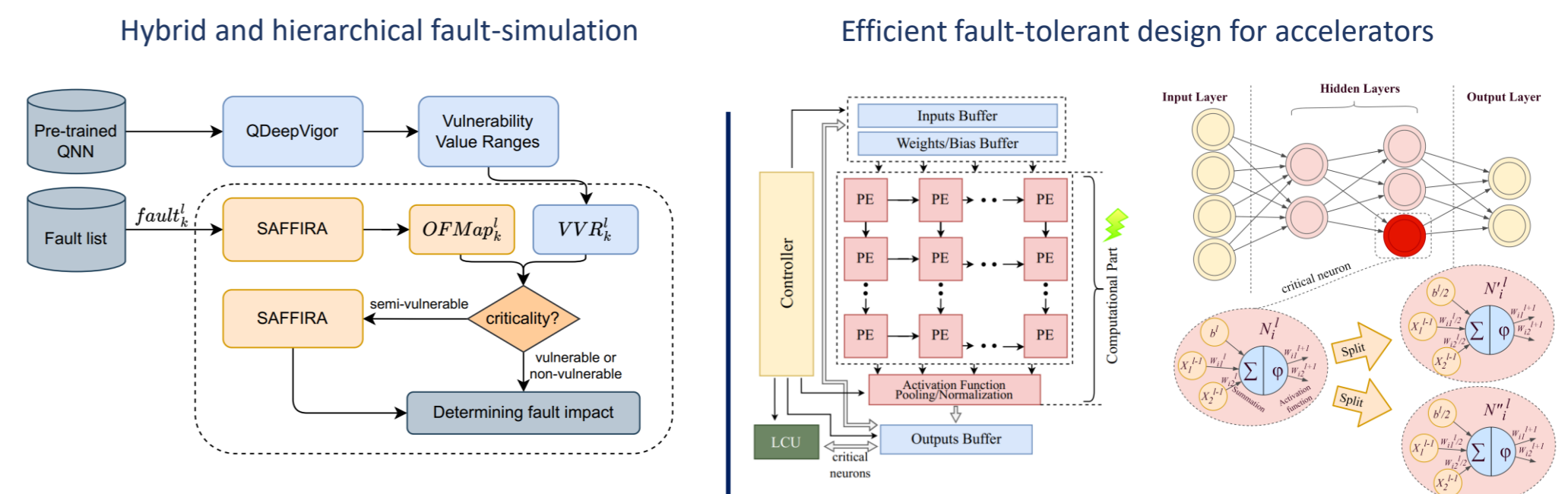
## Cont'd: Reliability Assessment for CNNs

- DeepVigor+:**
  - Addresses the scalability and provides accurate results within minutes
  - Open-source: [github.com/mhahmadilivani/DeepVigor](https://github.com/mhahmadilivani/DeepVigor)



Analysis method	Activations Analysis				Filters Analysis			
	Layer-wise	Data-unaware	Data-aware	Sampling DeepVigor+ 10%	Layer-wise	Data-unaware	Data-aware	Sampling DeepVigor+ 10%
VGG-11	74,863	1,562,657	71,173	4,596	75,351	2,141,913	66,934	1,038
VGG-16	117,992	1,839,889	106,513	10,283	123,266	3,588,834	112,151	1,476
ResNet-18-C	188,644	4,264,406	181,911	15,996	189,772	5,253,096	164,159	1,706
MobileNetV2	493,871	8,402,977	452,650	22,077	475,407	8,307,671	259,614	4,364
ResNet-18-I	191,205	5,407,659	189,325	21,680	190,896	5,358,315	167,447	2,178
ResNet-34	344,042	9,624,374	340,383	39,002	344,285	10,031,494	313,484	4,003

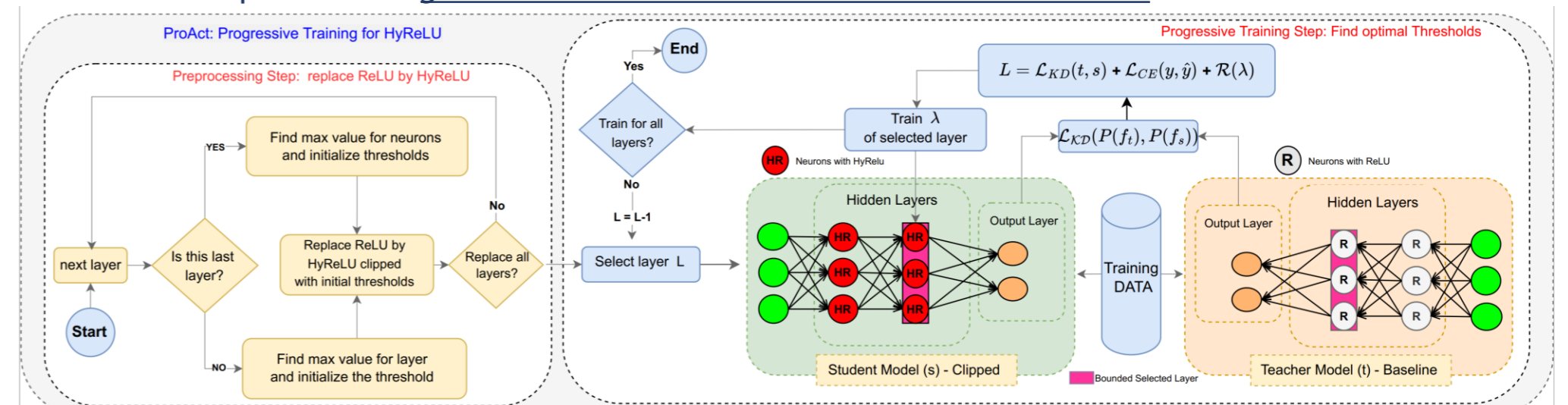
### Applications:



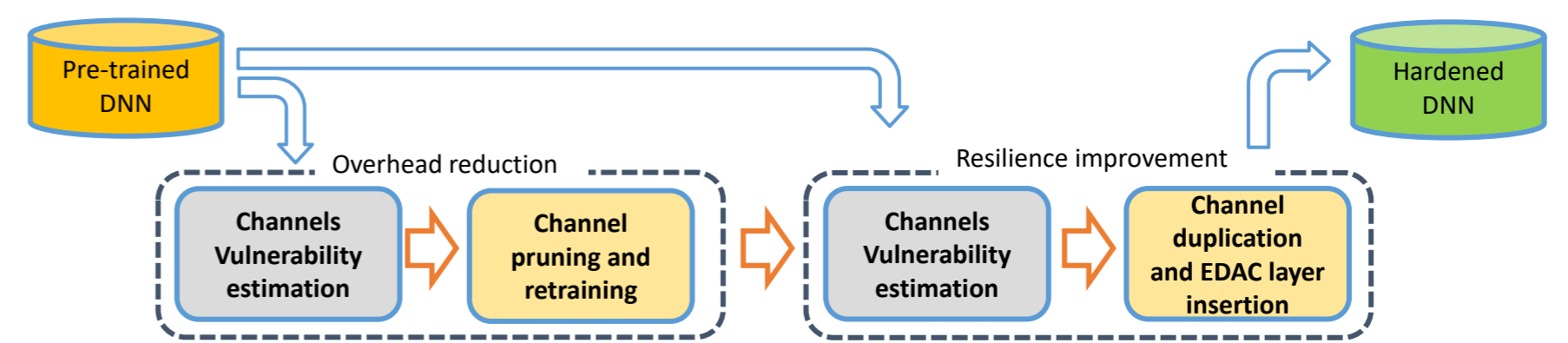
## Reliability Enhancement for CNNs

### ProAct and RReLU toolbox:

- Novel activation restriction method
- Introduces hybrid layer/neuron-wise activation functions with trainable thresholds
- Open-source: [github.com/hamidmousavi01/reliable-relu-toolbox](https://github.com/hamidmousavi01/reliable-relu-toolbox)

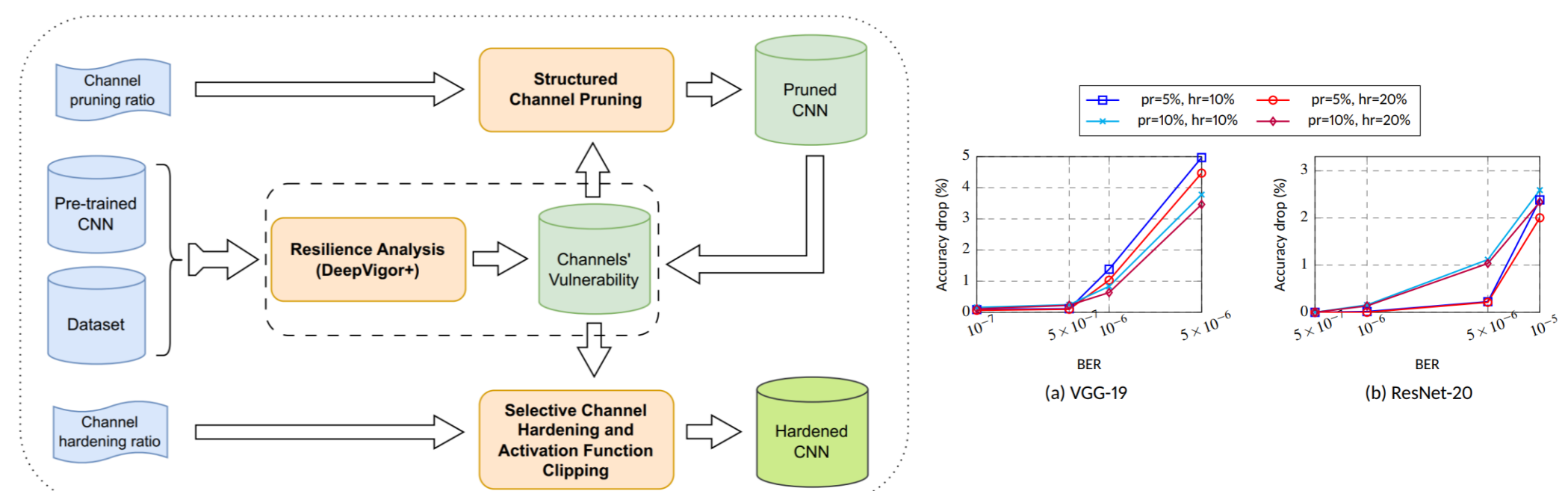


### Selective channel duplication and correction:



### SentinelINN framework:

- Integrates DeepVigor, vulnerability-aware channel pruning, selective channel duplication and correction, and range restriction
- Open-source: [github.com/mhahmadilivani/SentinelINN](https://github.com/mhahmadilivani/SentinelINN)

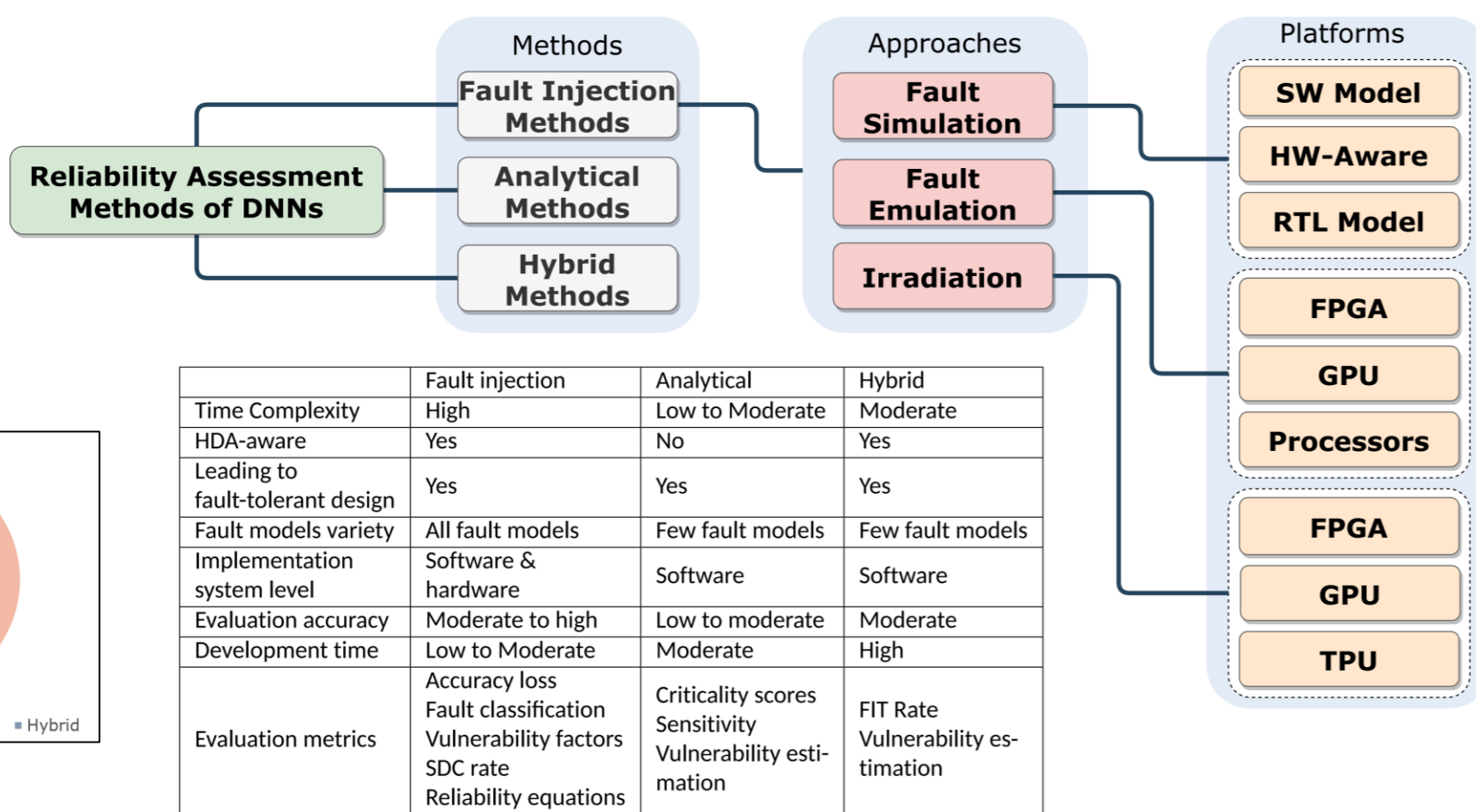


## Publications

- "A Systematic Literature Review on Hardware Reliability Assessment Methods for Deep Neural Networks", *ACM Computing Surveys*, Dec. 2023.
- "DeepVigor: Vulnerability Value Ranges and Factors for DNNs' Reliability Assessment", *ETS* 2023.
- "Enhancing Fault Resilience of QNNs by Selective Neuron Splitting", *AICAS* 2023.
- "Cost-Effective Fault Tolerance for CNNs Using Parameter Vulnerability Based Hardening and Pruning", *IOLTS* 2024.
- "DeepVigor+: Scalable and Accurate Semi-Analytical Fault Resilience Analysis for Deep Neural Network", *arXiv:2410.15742*, 2024.
- "ProAct: Progressive Training for Hybrid Clipped Activation Function to Enhance Resilience of DNNs", *arXiv:2406.06313*, 2024.
- "Analysis and Improvement of Resilience for Long Short-Term Memory Neural Networks", *DFTS* 2023.
- "Zero-Memory-Overhead Clipping-Based Fault Tolerance for LSTM Deep Neural Networks", *DFTS* 2024.

## Systematic Literature Review on Assessment Methods

- Includes and categorizes 139 related papers within 2017 to 2022.



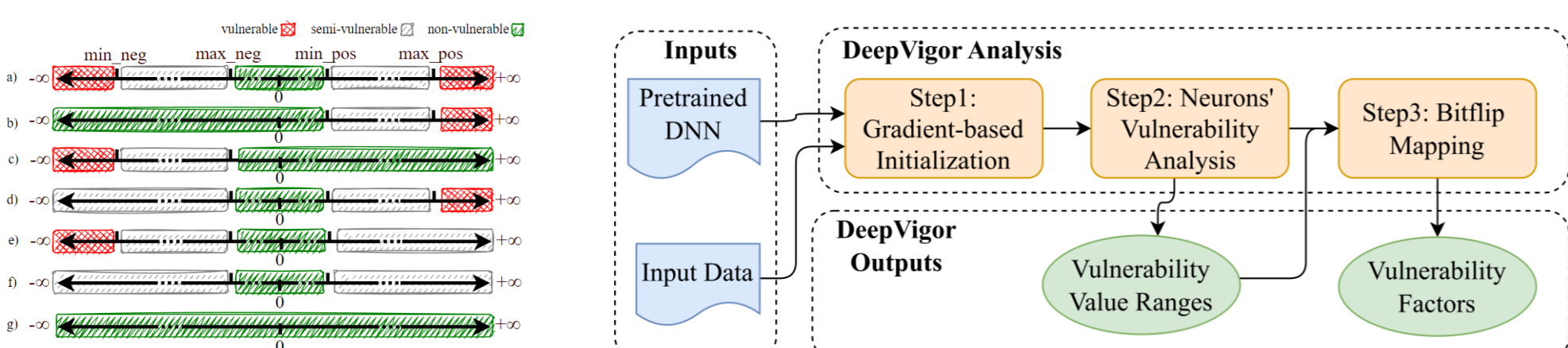
### Major identified gap:

- Need for scalable, fast, and accurate resilience analysis

## Reliability Assessment for CNNs

### DeepVigor:

- Establishes the concept of neurons' vulnerability ranges which indicates whether a fault in the output of neurons would lead to a DNN misclassification.
- Faster than FI, yet as accurate
- Provides Vulnerability Factors (VF) for layers, neurons, and bits within CNNs
- For floating-point and integer data types



DNN	True non-critical faults	True critical faults
MLP-sigmoid-mnist	99.985%~100%	100%
MLP-reLU-mnist	99.991%~100%	100%
LeNet-mnist	99.992%~100%	100%
LeNet-cifar10	99.956%~100%	100%
AlexNet-cifar10	99.973%~100%	99.955%~100%
VGG16-cifar100	99.950%~100%	99.972%~100%