Detection and Recovery Against Deep Neural Network Fault Injection Attacks Based on Contrastive Learning

Chenan Wang, Pu Zhao, Siyue Wang, Xue Lin
Northeastern University, Boston, MA, USA
{wang.chena,zhao.pu,wang.siy,xue.lin}@northeastern.edu

ABSTRACT

Deep Neural Network (DNN) models when implemented on executing devices as the inference engines are susceptible to Fault Injection Attacks (FIAs) that manipulate model parameters to disrupt inference execution with disastrous performance. This work introduces Contrastive Learning (CL) of visual representations i.e., a self-supervised learning approach into the deep learning training and inference pipeline to implement DNN inference engines with self-resilience under FIAs. Our proposed CL based FIA Detection and Recovery (CFDR) framework features (i) real-time detection with only a single batch of testing data and (ii) fast recovery effective even with only a small amount of unlabeled testing data. Evaluated with the CIFAR-10 dataset on multiple types of FIAs, our CFDR shows promising detection and recovery effectivenesses.

CCS CONCEPTS

• **Neural networks** → Security and privacy.

KEYWORDS

deep learning, fault injection attack, contrastive learning

ACM Reference Format:
Chenan Wang, Pu Zhao, Siyue Wang, Xue Lin. 2021. Detection and Recovery Against Deep Neural Network Fault Injection Attacks Based on Contrastive Learning. In *AdvML '21: Workshop on Adversarial Learning Methods for Machine Learning and Data Mining, August 15, 2021, Singapore (virtual).* ACM, New York, NY, USA, 5 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

Deep learning (DL) has succeeded in many application domains such as computer vision [3, 8] and natural language processing [2, 5]. Along with the prosperity of DL, its vulnerability under adversarial attacks has drawn significant attentions. For example, sophisticatedly crafted perturbations can be added onto clean images to produce adversarial examples [9, 10], the prediction of which by Deep Neural Networks (DNNs) will be erroneous, although the added perturbations are mostly imperceptible to humans.

Besides adversarial examples, Fault Injection Attacks (FIAs) present another category of adversarial attacks, which aim at DNN inference models when implemented on executing devices. In general, FIAs modify DNN model parameters, leading to malfunction of the inference models e.g., severely degradation in prediction accuracy or targeted misclassification of specified objects. Therefore, a DNN inference engine is subject to integrity violation caused by FIAs.

There are several state-of-the-art DNN FIAs proposed with diverse algorithms. Liu et al. proposed the first FIA with a heuristic algorithm i.e., Gradient Descent Attack (GDA) [6] that modifies DNN model parameters to classify specified inputs into wrong labels. Furthermore, Fault Sneaking Attack (FSA) [11] improved upon GDA with ADMM (Alternating Direction Method of Multipliers) based algorithms that set two constraints i.e., ℓ_0 or ℓ_2 norm of the parameter modifications, besides the specified misclassifications. On the other hand, He, Rakin, et al. proposed a FIA for quantized DNN models i.e., Bit Flip Attack (BFA) [4], which randomly picks model parameters and selects the most sensitive bit to flip. Then Progressive Bit Search (PBS) [7] extended BFA with cross-layer and intra-layer searches. Please note that both BFA and PBS target for tampering with DNN models through minimal modification efforts.

This paper aims to design DNN models with self-resilience under FIAs. We are the first to use Contrastive Learning (CL) towards this objective. Specifically, we use CL to obtain the DNN inference models. Comparing with conventional DNN training, CL enables our detection as well as recovery mechanisms against FIAs. With DNN inference models obtained by CL, we propose to observe the change of contrastive loss over single batches of testing data to detect potential FIAs. Since the detection mechanism does not require labeled data, it can be executed periodically without disruption on the normal DNN inference process. Whenever a FIA is detected, the recovery algorithm will be triggered to boost the accuracy performance close to that before the FIA. We consider two scenarios for the recovery process i.e., the labeled training data is available or only unlabeled testing data is available for recovery. In both scenarios, our proposed recovery algorithm can boost the accuracy significantly with only a small amount of data in a few number of epochs. We summarize our contributions as follows:

- The first CL based approach for FIA detection and recovery.
- A highly sensitive detection requiring a single batch of unlabeled data without disruption on normal inference process.
- A fast recovery algorithm that significantly boosts accuracy, even with only a small amount of unlabeled data.

2 PROPOSED CL BASED FIA DETECTION AND RECOVERY (CFDR)

This section introduces our main approaches by first fitting CL for our purpose, and then presenting our proposed detection and recovery mechanisms.

AdvML '21, August 15, 2021, Singapore (virtual)

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

2.1 Contrastive Learning and Preliminaries

Contrastive learning has been proposed as a self-supervised learning to reduce the requirement on the amount of labeled training data. For contrastive learning, we adopt the SimCLR method [1] that learns representations by maximizing agreement between differently augmented views of the same data example via a contrastive loss in the latent space. We will find in the following that (i) the contrastive loss is a key criterion in our FIA detection, and (ii) the relaxed requirement on the amount of labeled data by CL enables our effective recovery from FIA.

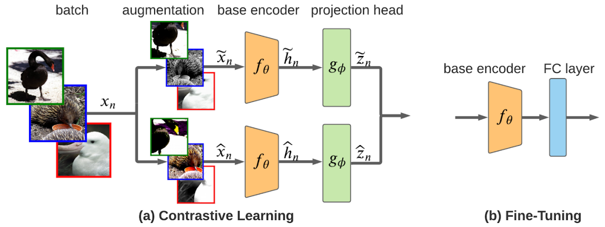

Figure 1: The SimCLR framework.

Figure 1 (a) and (b) present the two phases in the whole contrastive learning pipeline. We will first introduce the four components for Phase (a) Contrastive Learning:

- **A stochastic data augmentation module.** The module performs random combinations of data augmentation methods for each original data example x_n to generate a pair of correlated views of the same example, denoted by $\widehat{x}_n$ and $\widetilde{x}_n$, which are considered as a positive pair.
- **A base encoder f_θ.** It extracts image representation/embedding from augmented data examples. Following the SimCLR method, we also use ResNet (without the last fully-connected (FC) layer) as the base encoder to have
$$h_n = f_\theta(x_n). \tag{1}$$
- **A projection head g_ϕ.** It projects/maps high dimensional image representations to latent space with lower dimensionality, where the contrastive loss can be applied. It is a shallow multi-layer perceptron (MLP) with one hidden layer i.e.,
$$z_n = g_\phi(h_n). \tag{2}$$
- **A contrastive loss function.** It is designed for maximizing agreement between both image embeddings of the same data examples i.e., the positive pairs as follows:

$$Loss = -\sum_{n=1}^{N} \log \frac{\exp(\mathrm{sim}(\widetilde{\mathbf{z}}_n, \widehat{\mathbf{z}}_n)/\tau)}{\sum_{k=1,k\neq n}^{N} \exp(\mathrm{sim}(\widetilde{\mathbf{z}}_n, \widehat{\mathbf{z}}_k)/\tau) + \exp(\mathrm{sim}(\widetilde{\mathbf{z}}_n, \widetilde{\mathbf{z}}_k)/\tau)} \tag{3}$$

where $(\widetilde{\mathbf{z}}_n, \widehat{\mathbf{z}}_n)$ represents a positive pair, and $(\widetilde{\mathbf{z}}_n, \widehat{\mathbf{z}}_k)$ or $(\widehat{\mathbf{z}}_n, \widetilde{\mathbf{z}}_k)$ represents a negative pair. The $\mathrm{sim}(\cdot)$ function is the dot product between two ℓ_2 normalized vectors (i.e., cosine similarity).

With those above described components, the two phases in the whole contrastive learning pipeline are summarized as below:

(a) **Contrastive Learning Phase** as in Figure 1 (a). It trains the base encoder and the projection head with the contrastive loss. Only unlabeled data is needed in this phase.

(b) **Fine-Tuning Phase** as in Figure 1 (b). It fine-tunes on the FC layer using the regular cross entropy loss with a small amount of labeled training data, while the base encoder that is already trained in Phase (a) is not updated. Note that in our paper the base encoder plus the FC layer make a whole ResNet model architecture. The architecture in Figure 1 (b) is the finally obtained DNN inference model i.e., a SimCLR trained ResNet model.

2.2 Detection and Recovery Overview

With a DNN model obtained through the SimCLR method introduced in the previous section, now we introduce the overview of our proposed CL based FIA Detection and Recovery (CFDR) framework.

For the detection of FIAs, we propose to use the contrastive learning loss in Phase (a) as a key criterion to determine whether the DNN model is attacked by a FIA. We use the contrastive learning loss over a batch of data calculated with the clean (unattacked) model as the reference value. Note that the contrastive loss does not rely on labeled data, and therefore, the detection process can be co-executed during the normal DNN inference. Once a higher contrastive loss over a batch of testing data is observed, we determine that a FIA was conducted on the model. In summary, our detection mechanism features the real-time detection without interference with the normal inference execution.

Once a FIA is detected, our recovery algorithm will be triggered to boost the accuracy close to that before the FIA. We consider two scenarios for the recovery process. If the labeled training data is available for the recovery, we can recover a fault injection attacked model wherever the FIA is conducted on i.e., any layers of the DNN model. If only unlabeled testing data is available for the recovery, our recovery process can only address the cases where the FIA is conducted on any layers except the last FC layer. This is the limitation of our recovery mechanism.

2.3 Detection Mechanism

The detection of FIA is co-executed with the normal inference process with unlabeled testing data. Specifically, the contrastive loss over a batch of testing data is calculated with the same architecture as Figure 1 (a). Although this architecture is not directly used during inference, we can add the project head at the output of base encoder as the other path in parallel with the FC layer. This newly added path is for calculating contrastive loss simultaneouly with the inference execution. Therefore, we can implement real-time detection by comparing the contrastive loss with that of the clean model i.e., the reference value. If a relative large difference is observed, then the model is attacked. Besides the reference value, we set the fault tolerance parameter as the threshold for the difference. This fault tolerance parameter should be properly set to reduce the occurrences of false positives and false negatives. Furthermore, we can also use the contrastive loss over multiple batches for more accurate detection.

Algorithm 1: CL based FIA Detection and Recovery.

```
Data: The model to be detected, unlabeled testing data for
      detection, and a small amount of unlabeled/labeled
      data for recovery.
Result: The detection result and the recovered model if a
        FIA is detected.
Train a new DNN with the two-phase SimCLR method;
Compute the average contrastive loss $l_c$ for the clean model;
Set $l_c$ of the clean model as the reference value;
Set $\delta$ as the fault tolerance parameter;
Compute the contrastive loss $l_d$ for the model to be detected
  over a batch of unlabeled testing data;
if $|l_d - l_c| > \delta$ then
    the model is attacked;
    perform model recovery
    if labeled data is available then
        perform contrastive learning phase (a) for the model;
        perform fine-tuning phase (b);
    else
        perform contrastive learning phase (a);
    end
else
    the model is not attacked;
end
```

2.4 Recovery Mechanism

Once a FIA is detected, the recovery process is needed to boost the accuracy performance by retraining the model with a small amount labeled or unlabeled data. For the case that labeled training data is available, we perform both Phase (a) and Phase (b) for multiple epochs over the available training data. For the case that only unlabeled testing data is available for recovery, we perform only Phase (a).

Due to the small amount of labeled/unlabeled data for recovery, cautions must be used to avoid overfitting. Therefore, we use the following stopping criteria:

(1) The training loss is less than or equal to a reference value. For the contrastive loss in Phase (a) or the cross entropy loss in Phase (b), we use their counterpart on the clean model as the reference value. **OR**
(2) The training loss stops decreasing. **OR**
(3) The total epoch number reaches a certain value e.g., 30.

3 EXPERIMENTS

3.1 Experimental Setup

We evaluate our CFDR framework using a ResNet-18 model trained with the SimCLR framework on CIFAR-10 dataset. We adopt a batch size of 64 throughout the whole process i.e., SimCLR training, detection and recovery. For the original SimCLR training, Phase (a) uses 1,000 epochs and Phase (b) uses 100 epochs, i.e., the same setting as the SimCLR codes.

For detection, the contrasive loss over a single batch of testing data is sampled 1,000 times to obtain the detection results in Section 3.2. For recovery, we assume only 512 images are available for both the unlabeled and labeled data cases.

For evaluation, we adopt four types of FIAs i.e., PBS [7], FSA ℓ_0 [11], FSA ℓ_2 [11], and GDA [6]. We use attacked models by the above-mentioned four attacks in various settings to evaluate our CFDR framework.

3.1.1 *PBS.*

Progressive bit search (PBS) [7] performs in-layer search and cross-layer search. In-layer search finds the most vulnerable bits from a layer; cross-layer search finds the most vulnerable layer with in-layer search. The goal of PBS is to tamper with the network, more precisely, degrading the top-1 accuracy of attacked network below 11%. For PBS, we adopt the default setting of [7], i.e. the hacker can change all parameters in the attacked layer. Normally, single PBS run tampers with the network below 11%.

3.1.2 *FSA.*

Fault sneaking attack (FSA) [11] uses efficient ADMM (alternating direction method of multipliers) algorithms to modify model parameters, so that the model would make wrong predictions. In the experiment, we found out that FSA usually changes almost all parameters in a layer. For FSA, we set S=5 and R=20, i.e. we modify 5 images out of 20 images. 5 images are misclassified to wrong labels while 15 images keeps the original correct labels.

3.1.3 *GDA.*

Similar to FSA, Gradient descent attack (GDA) [6] is more straightforward. It gradually modifies parameters with gradient information to enlarge the predictions of a specific class, leading to incorrect predictions. At the same time, they use an ℓ_1-norm regulator to limit the parameter modifications. Different from the original setting, we do not perform modification compression, since its iteration is inefficient. Besides, we use an ℓ_2 regulator to restrict model modifications.

3.2 Results on Detection

Figures 2, 3, 4, and 5 represent the detection effectiveness of our CFDR framework against PBS, FSA ℓ_0, FSA ℓ_2, and GDA attacks. For each attack type, we conducted FIA multiple times, acting on different layers of the DNN model. The number of parameters in the modified layers of each attack instance is the x-axis. The contrastive loss is sampled 1,000 times over single batches of testing data. As can be observed from the figures, the contrastive loss is well seperated from that of the clean model, demonstrating the effectiveness of our detection mechanism.

3.3 Results on Recovery

Table 1 presents the recovery effectiveness of our CFDR framework

For each FIA type, we generate various attacked models by perturbing different layers with different number of parameters. As shown in Table 1, the accuracy after attack suffers from obvious degradation. But after we successfully detected the attacks, we can perform model recovery with labeled or unlabeled data. After the recovery, we are able to improve the accuracy on the test set. For models with less perturbed/attacked parameters, the accuracy can be restored to a typical accuracy (around 87%) for ResNet-18 on CIFAR-10 with only a few data. For models with more perturbed/attacked parameters (e.g., 1180672 parameters), the accuracy can be improved to 45% for GDA with labeled data, which is still

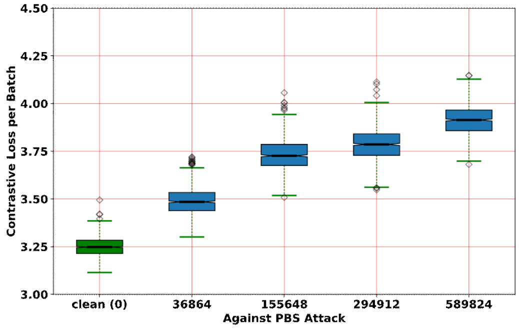

Figure 2: Detection effectiveness by box plot when different number of parameters are modified by the PBS

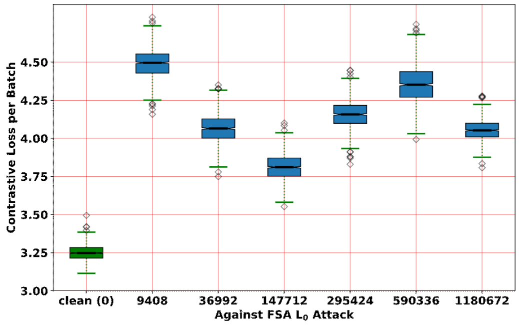

Figure 3: Detection effectiveness by box plot when different number of parameters are modified by the FSA ℓ_0

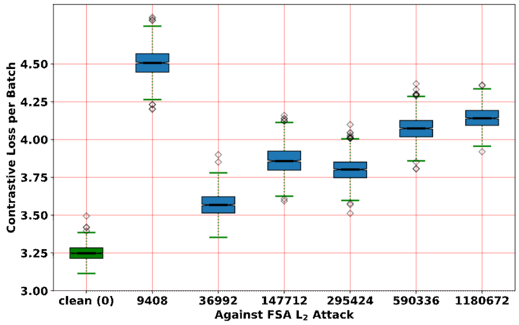

Figure 4: Detection effectiveness by box plot when different number of parameters are modified by the FSA ℓ_2

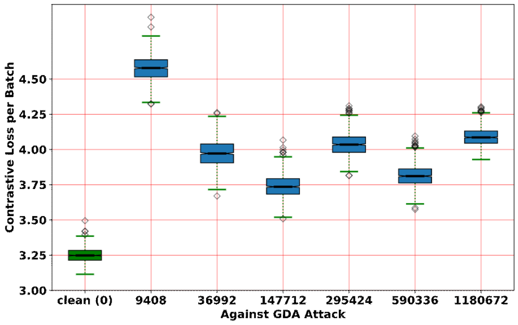

Figure 5: Detection effectiveness by box plot when different number of parameters are modified by the GDA

Table 1: Recovery effectiveness

Attack	total # of param. in the attack -ed layer(s)	after attack acc.	unlabeled recovery acc.	epochs	labeled recovery acc.	epochs
PBS	36864	10.00	88.71	6	88.89	7
PBS	155648	10.16	87.72	15	87.64	22
PBS	294612	13.25	87.36	15	87.34	5
PBS	589824	10.00	80.34	12	81.05	5
FSA ℓ_0	9408	12.57	85.15	30	85.57	9
FSA ℓ_0	36992	10.07	88.09	20	88.11	15
FSA ℓ_0	147712	10.47	87.68	24	88.22	7
FSA ℓ_0	295424	14.43	79.63	18	80.82	5
FSA ℓ_0	590336	17.03	65.84	15	68.52	10
FSA ℓ_0	1180672	11.86	33.45	21	40.41	10
FSA ℓ_2	9408	11.17	85.23	30	85.65	9
FSA ℓ_2	36992	13.6	88.50	10	88.41	3
FSA ℓ_2	147712	11.04	87.53	22	87.93	5
FSA ℓ_2	295424	15.22	83.70	11	84.19	5
FSA ℓ_2	590336	10.95	70.57	20	72.37	5
FSA ℓ_2	1180672	12.84	28.42	19	35.07	15
GDA	9408	46.79	82.77	30	82.93	9
GDA	36992	83.42	88.08	9	88.33	9
GDA	147712	84.72	87.62	9	87.87	24
GDA	295424	55.89	80.41	17	80.86	9
GDA	590336	58.69	79.38	27	79.92	10
GDA	1180672	25.41	32.79	30	45.36	30

smaller than the normal accuracy with a large gap. This demonstrates the limitations/difficulties for recovery to restore if too many parameters (e.g. more than 1 billion) are perturbed. We also notice that the recovery with labeled data can usually achieve higher accuracy than the recovery with unlabeled data, demonstrating that more information with labels can help with the training.

4 CONCLUSION

This work introduces Contrastive Learning (CL) of visual representations into DL training and inference pipeline to implement DNN inference engines with self-resilience under FIAs. Our proposed CL based FIA Detection and Recovery (CFDR) framework features (i) the first CL based approach for FIA detection and recovery; (ii) a highly sensitive detection mechanism requiring a single batch of unlabeled testing data without disruption on the normal inference process; and (iii) a fast recovery algorithm that significantly boosts accuracy performance even only with unlabeled testing data. Evaluated with the CIFAR-10 dataset on multiple types of FIAs, our CFDR shows promising detection and recovery effectivenesses.

ACKNOWLEDGMENTS

This work is supported by National Science Foundation (NSF) under Award # CNS-1929300.

REFERENCES

[1] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. 2020. A simple framework for contrastive learning of visual representations. In *International Conference on Machine Learning (ICML)*. PMLR, 1597–1607.

[2] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805* (2018).

[3] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 770–778.

[4] Zhezhi He, Adnan Siraj Rakin, Jingtao Li, Chaitali Chakrabarti, and Deliang Fan. 2020. Defending and harnessing the bit-flip based adversarial weight attack. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 14095–14103.

[5] Geoffrey Hinton, Li Deng, Dong Yu, George E Dahl, Abdel-rahman Mohamed, Navdeep Jaitly, Andrew Senior, Vincent Vanhoucke, Patrick Nguyen, Tara N Sainath, et al. 2012. Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups. *IEEE Signal Processing Magazine* 29, 6 (2012), 82–97.

[6] Yannan Liu, Lingxiao Wei, Bo Luo, and Qiang Xu. 2017. Fault injection attack on deep neural network. In *2017 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*. IEEE, 131–138.

[7] Adnan Siraj Rakin, Zhezhi He, and Deliang Fan. 2019. Bit-flip attack: Crushing neural network with progressive bit search. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*. 1211–1220.

[8] Christian Szegedy, Sergey Ioffe, Vincent Vanhoucke, and Alexander Alemi. 2017. Inception-v4, inception-resnet and the impact of residual connections on learning. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, Vol. 31.

[9] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna Estrach, Dumitru Erhan, Ian Goodfellow, and Robert Fergus. 2014. Intriguing properties of neural networks. In *International Conference on Learning Representations (ICLR)*.

[10] Kaidi Xu, Sijia Liu, Pu Zhao, Pin-Yu Chen, Huan Zhang, Quanfu Fan, Deniz Erdogmus, Yanzhi Wang, and Xue Lin. 2019. Structured Adversarial Attack: Towards General Implementation and Better Interpretability. In *International Conference on Learning Representations (ICLR)*.

[11] Pu Zhao, Siyue Wang, Cheng Gongye, Yanzhi Wang, Yunsi Fei, and Xue Lin. 2019. Fault sneaking attack: A stealthy framework for misleading deep neural networks. In *2019 56th ACM/IEEE Design Automation Conference (DAC)*. IEEE, 1–6.