Selective Hardening of CNNs based on Layer Vulnerability Estimation

Cristiana Bolchini, Luca Cassano, Antonio Miele, Alessandro Nazzari
Dipartimento di Elettronica, Informazione e Bioingegneria Politecnico di Milano, Italy
{first_name.last_name}@polimi.it

***Abstract*—There is an increasing interest in employing Convolutional Neural Networks (CNNs) in safety-critical application fields. In such scenarios, it is vital to ensure that the application fulfills the reliability requirements expressed by customers and design standards. On the other hand, given the CNNs extremely high computational requirements, it is also paramount to achieve high performance. To meet both reliability and performance requirements, partial and selective replication of the layers of the CNN can be applied. In this paper, we identify the most critical layers of a CNN in terms of vulnerability to fault and selectively duplicate them to achieve a target reliability vs. execution time trade-off. To this end we perform a design space exploration to identify layers to be duplicated. Results on the application of the proposed approach to four case study CNNs are reported.**

***Index Terms*—Convolutional Neural Networks, Functional Vulnerability Factor, Reliability Analysis, Selective Hardening.**

I. Introduction

Convolutional Neural Networks (CNNs) are widely adopted to provide perception functionalities in several application domains, including safety-critical ones. An example is the Advanced Driver Assistance System (ADAS) in the automotive scenario [1], where CNNs detect track lanes, interpret road signs and traffic lights, and identify pedestrians and obstacles [2], [3]. Based on such observations, subsequent planning modules take trajectory and control decisions.

The design of digital systems in safety-critical applications fields is regulated by standards, e.g., the ISO 26262 for automotive [4]. In the same scenario, the functionalities provided by an ADAS are regulated by the Society of Automotive Engineers (SAE). Both standards require high reliability and fault detection/management mechanisms to be exposed by the system. Among the possible causes of failures of digital systems, it has to be mentioned that soft errors, e.g., Single Event Upsets (SEUs) [5], may interfere with electronic systems also at the ground-level [6]. A rate of two transient faults every thousand billion hours, on average, has been estimated. Although this fault rate may appear negligible, if we take into account that in 2019 the number of cars traveling in Europe has been 268 millions [7], the estimate of faults per car would be one every 3.7 hours, which indeed may be a concern.

In this application scenario, though, classical redundancy-based software-level fault detection and mitigation techniques, such as Duplication with Comparison (DWC) or Triple Modular Redundancy (TMR), may not be affordable when applied to CNNs due to the excessive performance overheads [8]. However, applications exploiting deep learning generally expose an intrinsic degree of fault resiliency [9]. It is therefore possible to analyze the effects of faults in the architecture and how they interfere with the top-most application, to tailor the hardening scheme. To limit the overheads, recent studies (e.g., [8], [10], [11]) proposed selective duplication at the granularity of layers or feature maps, duplicating only the CNN layers (or feature maps) that most contribute to the failure rate of the system. Different strategies are used to estimate layers' vulnerability and to determine which ones to harden.

We tackle the same problem, CNN robustness against SEUs, by exploiting selective hardening of the CNN layers based on the estimation of their vulnerability, introducing two main contributions: i) vulnerability/robustness is estimated with respect to a usability-based concept [12], rather than the classical correct/incorrect one, and ii) a Design Space Exploration (DSE) approach for exploring the relevant selective hardening alternatives, rather than using heuristic-based policies. The former element allows us to better exploit the inherent nature of CNN-based application context, focusing on whether the downstream system is able to correctly carry out its task even when CNN output is corrupted. The latter aims at allowing the designer to choose among the most relevant alternatives, leveraging reliability and performance degradation.

We estimate the Layer Vulnerability Factor (LVF) [8] of the layers of the CNN, by means of CLASSES [13]. Based on this estimation of the fault susceptibility and on the profiled execution times, the DSE explores the solution space by evaluating the overall reliability and execution time of each hardened solution and identifies the Pareto points representing the most interesting ones. We applied the proposed selective hardening methodology to four CNNs and we show that it is possible to significantly reduce the CNN execution time with an extremely reduced decrease in reliability.

The remainder of this paper is organized as follows. Section II introduces the necessary background and a brief review of related work. Section III describes the proposed analysis and hardening method, and Section IV reports its application to the selected case studies. Finally, Section V draws conclusions.

II. Background and Related work

A. The Considered Working Scenario

The scenario we envision is the one of cyber-physical systems, such as the one of an autonomous rover, where

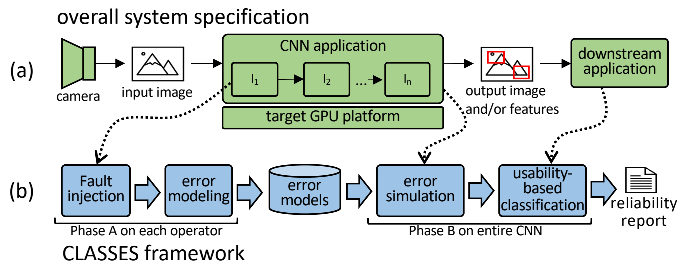


Figure 1. The considered system (a) and the CLASSES framework (b).

part of the computation includes image processing tasks. The overall system uses a camera to take images of the surrounding environment, that are then processed by a CNN, as in Figure 1(a). The CNN analyzes the captured images to detect, classify and/or locate features that are possibly used to enhance the input image. The output data is then used by a downstream control application, to take decisions.

CNNs are internally organized in a pipeline of layers, each one performing one or multiple operations, such as convolutions, batch normalizations or poolings. We propose a software-based selective hardening of the CNN to limit performance degradation overheads, that works at the granularity level of the single CNN layer. For the sake of simplicity, we assume each CNN layer to contain a single operator, therefore both robustness and duplication considerations are directly referred to the layer itself. However, the approach can be easily extended to the general case of multiple operators within a single layer, and we will comment on the overall impact.

The adopted fault model is the Single Event Upset, which leads to a transient computational error at *inference time*. A transient error could either have no effect, get detected by the OS or remain undetected and corrupt the output data, a so-called **Silent Data Corruption (SDC)**. We here focus on the latter class, SDC, being interested in improving the resiliency of the CNN-based perception module.

Due to their data-intensive nature, CNNs are generally accelerated onto **Graphic Processing Units (GPUs)** or on custom designed/configurable hardware, and here we refer to the former. The choice of the underlying hardware platform affects the effects of faults on the running application. In particular, when working at the application level to simulate/analyze the errors produced by a fault in the CNN computation, the hardware architecture is a fundamental aspect. In the present study, the error models adopted and used in the CLASSES framework (depicted in Figure 1(b)) are those associated with the GPU hardware platform.

Finally, we adhere to the **usable/unusable paradigm** rather than the correct/incorrect one, so that the CNN corrupted output is actually considered so only if the downstream application will not be able to exploit it, and to this end we rely on the CLASSES framework [13] to carry out the layer robustness analysis, as explained in the following.

B. Related work

Selective (or partial) hardening is not a new strategy and it has been investigated in the past as a viable solution to limit reliability-related overheads, working either at hardware or software level. Sometimes the strategy is used in a best-effort fashion, getting the most from the reliability point of view, based on the available resources. In most cases, the selection of the elements to be hardened is arbitrary, based on the designer's considerations on the criticality of the components and/or on the available resources. In the context of CNN resiliency, there are a few recent works, among which [8], [11], [14]. In the first two works, an analysis is performed to identify the layers of the CNN that have more impact on its correct final output, being implemented on an FPGA, and on a GPU, respectively. In [8] the concept of Layer Vulnerability Factor is introduced to characterize the relevance of the errors in the layers with respect to the correctness of the CNN output, the same we here adopt. The estimated LVFs are directly used by the designer to decide what layers to replicate, in a selective hardening strategy, without any DSE.

Selective duplication based on the vulnerability of the CNN elements is presented in [11], which identifies and statically protects the most vulnerable feature maps in a CNN. The strategy aims at achieving the desired reliability, limiting the impact of the hardening as much as possible, and the designer is therefore unable to select the trade-off they deem most interesting. In the future we plan to explore the opportunity to port our DSE to such a context to compare the benefits of working on different CNN elements.

As far as the approach at estimating the LVF for our DSE is concerned, the mentioned studies employ *Radiation Testing (RT)*, *Fault Injection (FI)* and *Fuctional Error Simulation (FES)* solutions. RT is able to accurately reproduce the effects of radiation in the circuit but it is extremely expensive and it does not provide enough internal controllability and observability. FI, e.g., [15], [16], [17], provides high controllability and observability of the experiments, as well as high accuracy. However, FI techniques generally require a long time and effort to be integrated within the system under analysis and they can generally be applied very late in the design process. Moreover, FI requires the execution of extensive experimental campaigns in order to collect a statistically relevant amount of corrupted data since faults may either be not activated or produce no observable error.

The last family of techniques is *Application-Level FES*, e.g., [18], [19], [20], whose advantages are: i) shorter development and deployment times, ii) ease of use and availability early during the design process of the application under analysis, and iii) no fault activation issues. On the other hand, the accuracy of the analysis carried out by FES engines strongly depends on the accuracy of the adopted error models.

The most representative FES engine is TensorFI [18] that is an error simulator for CNNs integrated in the TensorFlow framework. TensorFI emulates the effects of faults affecting the execution of the operators of the CNN. A similar strategy has been adopted in other FES engines, i.e., [19], [20], integrated into the Pytorch and Caffe frameworks, respectively. Other methodologies, like [21], [22], only support the corruption of the operator weights. However, the error models adopted by TensorFI (at the basis of the work in [11]) have some limitations as discussed in [13], therefore we adopted CLASSES. As shown in Figure 1(b), it relies on an architecture-level FI applied on each single CNN operator to define a set of accurate error models that are representative of the effects of the HW-level faults affecting the underlying GPU. These error models are then integrated into a FES environment to carry out an early, fast and still accurate reliability evaluation of the entire CNN w.r.t. usability.

III. The Proposed Methodology

The method we propose first performs the reliability analysis of the CNN layers, then explores the DSE associated with the different selective hardening alternatives, as detailed in the following.

A. The Reliability Analysis

The proposed analysis method relies on the LVF metric [8], defined as the probability of a fault injected in a specific layer to affect the final CNN output. Starting from the results of a fault injection campaign consisting of $\#exps$ experiments targeting a specific layer i, LVF is computed as:

$$LVF_i = \frac{\#sdc_i}{\#exps} \tag{1}$$

where $\#sdc_i$ is the number of SDCs, i.e., faults injected in the layer whose effect is propagated to the final CNN results. To align the metric to the usability paradigm, we here replace $\#sdc_i$ with $\#\overline{u_i}$, that is the number of faults causing an *unusable* result (instead of the *incorrect* result). We identify our LVF metric related to usability as $\widetilde{LVF}_i$, computed as:

$$\widetilde{LVF}_i = \frac{\#\overline{u_i}}{\#exps} \tag{2}$$

The corresponding *robustness* of the layer is:

$$R_i = 1 - \widetilde{LVF}_i = 1 - \frac{\#\overline{u_i}}{\#exps} \tag{3}$$

Based on the CLASSES workflow, we split the computation of R_i in two steps. First, architecture-level fault injection is used to analyze the susceptibility of each operator i[1] to faults. More precisely, results of the preliminary fault injection campaigns used for error modeling can be used to retrieve the number of experiments $\#errors_i$ where an error has been observed in the output tensor among the overall $\#faults_i$ runs. Thus, the susceptibility of operator i, s_{err_i}, is computed as:

$$s_{err\_i} = \frac{\#errors_i}{\#faults_i} \tag{4}$$

In the second phase, we collect the effect of erroneous tensors produced by the layer i (executing operator i) on the final output of the CNN, with respect to its usability. We inject $\#sim\_errs_i$ errors on the output of the considered layer to collect the number of *usable* results $\#u_i$, notwithstanding the error in the tensor output.

[1]Given the fact we here adopt the one layer-one operator reference, we use index i for both the operator and the layer.

These two contributions are combined to compute the layer robustness, that is the probability of the CNN output to be usable when a fault affects the GPU executing the application, such that layer i could be affected, and it is formulated as:

$$R_i = (1 - s_{err\_i}) + s_{err\_i} \cdot \frac{\#u_i}{\#sim\_errs_i} \tag{5}$$

where the first part of the formula represents the percentage of cases where the fault did not cause an error on the layer output, and the second part represents the percentage of cases where the fault led to a final usable result. Note that here $\#u_i$ refers to the number of executed error simulations.

Once we have computed R_i for each layer i, the robustness of the entire CNN can be computed, R_{CNN}. All layers are executed in sequence on the same GPU, the probability of the fault to affect the i-th layer can be approximated with the portion of time needed to run it w.r.t. running the entire CNN. In the single fault assumption, if a fault occurs while running layer i, the execution of other layers is fault-free. Therefore the conditional probability of layer i performing correctly if a fault occurs is R_i. Thus, according to the law of total probability and the theorem of Bayes, R_{CNN} can be computed as:

$$R_{CNN} = \sum_{i \in CNN} R_i \cdot \frac{t_i}{t_{CNN}} \tag{6}$$

being t_{CNN} and t_i the profiled execution time of the overall CNN and of layer i, respectively.

B. The Selective Hardening Strategy

R_{CNN} measures the intrinsic resiliency of the CNN against faults, i.e., its capability to output *usable* elaborations also in presence of faults. The aim of this paper is to define a selective hardening technique maximizing such a value while limiting the introduced performance overhead.

We here adopt the DWC applied at the granularity of the single layer. More in details, two instances of the same layer are executed in sequence and the produced results are then checked by an ad-hoc additional checking layer, executed on the GPU as well. The proposed scheme can detect errors in the single layer run; being it in a software context, a re-execution of the single layer can be triggered to recover from the error.

The duplication of the layer improves the CNN reliability, w.r.t. fault detection properties, incurring in a performance degradation PD_i related to the additional execution of the layer t_i, plus the additional checking layer t_{c_i}. Therefore, the replication of layer i is characterized by:

$$R_i = 1 \tag{7}$$

$$PD_i = t_i + t_{c\_i} \tag{8}$$

Do note that being the fault a rare event, the overhead of error correction is negligible, as commented in [12].

The application of the DWC to the entire CNN constitutes a boundary solution, characterized by the highest overall robustness and performance degradation and does not exploit its inherent resiliency. We propose to selectively harden the CNN, by applying the DWC only to a subset of the layers H_CNN

(while leaving the remaining $\overline{H}\_CNN$ layers unprotected), thus achieving a partial robustness with a limited performance degradation. The selectively hardened solution is characterized by the following two figures of merit measuring robustness and execution time, respectively:

$$R_{SH\_CNN} = \sum_{i \in \overline{H}\_CNN} R_i \cdot \frac{t_i}{t_{CNN}} + \sum_{i \in H\_CNN} \frac{t_i}{t_{CNN}} \quad (9)$$

$$t_{SH\_CNN} = t_{CNN} + \sum_{i \in H\_CNN} PD_i \quad (10)$$

To support the selective hardening strategy, we adopt a multi-objective DSE process that explores the solution space of possible combinations of replicated layers evaluated w.r.t. the two above figures of merits. The process returns a Pareto front where the best solutions lie; the designer has therefore the opportunity to select the most interesting one according to the given constraints (e.g., a deadline on the execution time). The solution space is not characterized by relevant constraints, thus a straightforward multi-objective genetic algorithm engine has been adopted for a fast DSE execution. In particular, the solution has been encoded in a chromosome where each gene is a binary variable representing whether a layer is hardened or not; then the well-known NSGA-II genetic algorithm has been adopted by using classical single point crossover and mutation operators. Given the limited effort to perform the DSE, we perform a systematic exploration of the a richer set of solutions w.r.t. the work in [9], which only identifies a single, good hardened solution using a LVF-based heuristic. Indeed, our proposal also identifies such a solution among the numerous alternative ones.

IV. Experimental Evaluation

The proposed analysis and DSE methodology has been implemented in a Python script interfaced with CLASSES prototype. We conducted an experimental campaign aimed at assessing the effectiveness of the proposed approach. For the experiments we selected four CNNs, performing different perception functionalities, taken from the set of applications considered in the experimental campaign performed in [18]. In particular we employed CNN implementations in TensorFlow and datasets provided by the developers. Experimental tests have been performed on the NVIDIA Jetson TX2 board.

A. The considered CNN applications

Comma's AI Steering Wheel Model performs steering angle detection for autonomous driving applications. Its structure is composed of three subsequent convolution layers followed by two subsequent bias add layers. According to the analysis performed in [18], we defined a usability-based classifier tagging as usable any processing result differing from the expected predicted angle by less than $5°$.

PilotNet is another automotive application for steering angle detection. Its structure is composed of five subsequent convolutional layers followed by four subsequent fully connected layers and a final multiplication operator. We adopted here the usability-based classifier previously presented for the Comma's AI Steering Wheel Model.

CIFAR-10 is a CNN devoted to image classification into 10 different classes. CIFAR-10 is composed by two convolution + batch normalization layers followed by a maxpooling layer and then four additional convolution + batch normalization layers and again a maxpooling layer; the last layers are two fully connected and a softmax. The usability-based classification is straightforward: the corrupted output is usable if the predicted class is the same as the class predicted by fault-free CNN.

Vgg11 performs image classification on traffic signs. Vgg11 has a structure similar to CIFAR-10: two convolutional layers followed by a batch normalization layer and then four convolutional layers with again a batch normalization; the structure is then completed by three subsequent fully connected layers. The usability-based classifier works as the one defined for the CIFAR-10 CNN.

B. Baselines

We selected four alternative solutions to compare against:

- the plain CNN with no redundancy;
- an application-level DWC, where the entire CNN is duplicated and only the final result is checked;
- a layer-level DWC where every layer is duplicated and each intermediate result is checked; and
- the hardened solution identified by means of the heuristic proposed in [8].

The plain solution only relies on the CNN intrinsic degree of fault resiliency, as often seen in literature. The two full DWC schemes provide 100% fault detection and are characterized by different execution times, because of the presence of one or more checking layers. This also affects the re-execution policy, upon error detection: in the former case the entire CNN is re-executed, in the latter, only the corrupted layer. The heuristic proposed in [8] sorts layers according to LVF and then proceeds by duplicating them incrementally until a satisfactory solution is found. However, the paper does not define any stop criterion for the process. This solution is indeed included in the ones identified by the DSE.

C. Experimental results

We used CLASSES to compute s_{err_i} and R_i for all layers considered in the CNNs. It is worth mentioning that the computation of s_{err_i} is application independent and once collected, they are always available in a repository, for the analysis of any CNN. We report in Table I the results; they are later used to compute the robustness of the plain CNN and of the selectively-hardened versions. For the sake of space, the table only reports data belonging to the Comma CNN. Moreover, the table reports the execution time of the layer and of the checker that should be instantiated in case the layer is chosen for replication. It can be observed that s_{err_i} is always below 50% and in several cases also around, or even below, 30%. We may therefore argue that basic operators are already intrinsically resilient to faults. Moreover, the last layer of the CNN is, as expected, the most critical one; the CNN fails in

Table I
EXPERIMENTAL RESULTS FOR THE COMMA CNN

Layer	s_{err_i}	R_i	Execution Time (ms)	Checker Time (ms)
Convolution1	0.43	0.81	76.253	0.394
Convolution2	0.43	0.74	81.231	0.318
Convolution3	0.43	0.73	85.208	0.322
BiasAdd1	0.47	0.95	0.158	0.389
BiasAdd2	0.47	0.08	0.151	~ 0

92% of the cases when the layer produces a wrong output. On the other hand, the robustness of the CNN against errors affecting the other layers is much higher, up to 95% for the first BiasAdd, thus confirming that the overall CNN internally absorbs erroneous values.

The values computed with these two preliminary experiments are then used as input of the DSE process. The results for the four considered CNNs are reported in Figure 2, where we plot the robustness and execution time of the explored solutions (the legend in only reported in the first plot for the sake of readability). In the plots, the red dots represent the overall explored solution space while the Pareto front and the baselines are shown in different colors.

As expected the plain solution and the layer-level DWC represent the two boundary points of the Pareto front; the former having minimum execution time and robustness, while the latter achieving a complete fault coverage while incurring in the maximum performance overhead. As a first consideration This analysis confirms what emerges in literature, that although inherently resilient, the plain CNN does not exhibit a high robustness. More precisely, PilotNet reaches a 96%, however Comma and Vgg11 are characterized by a 84% and 64%, respectively. On the other side of the plot, we can notice that the layer-level DWC an the application-level one present very similar characteristics; the reason is that the checking time is quite limited, thus checking the intermediate outputs of each layer does not incur in a relevant performance overhead. As a consequence, layer-level DWC is more beneficial: indeed, it allows a smaller recovery time since the actual layer affected by the error can be identified and re-executed. The only exception to these considerations is represented by Vgg11 where application-level DWC is relevantly faster since the execution times of the various layers is much shorter than in the other considered CNNs. As a consequence, all the intermediate checking introduced by layer-level DWC represent a much more relevant overhead.

Between the two boundary solutions there is the selective hardening solution space. The size of the overall solution space highly varies depending on the number of layers in the CNN; indeed some of he considered applications (Comma and PilotNet) are quite small. Future work is devoted to the analysis of a much more complex case study, that is Yolo, comprising approximately 280 layers, and therefore generating a much broader space.

The proposed DSE identifies a rich Pareto front, where designers can select the most interesting trade-off based on their needs or on any imposed constraint, e.g., on the system performance. For example, when considering CIFAR-10 or PilotNet, it can be observed that it is possible to reduce the execution time of about 150ms reducing only about 1% robustness w.r.t. the per-layer DWC scheme. The solutions identified by the heuristic proposed in [8] either belong to the elements of the Pareto front identified by our DSE methodology, or, sometimes, they are sub-optimal points very close to it. Layers sorting according to the identified LVF and then selectively duplicating the layers based on this sorting is a quite effective strategy to identify a reduced set of relevant solutions in a very short execution time (about one second for the considered CNNs). Yet, the genetic algorithm performing the proposed DSE runs in almost one minute, an affordable time, also considering that the entire Pareto front is returned.

As a final note, we want to mention that the generalization of the CNN model where multiple operators are executed within a single layer would only affect the computation of R_i. Indeed, in the mentioned generalized CNN models R_i should be computed by taking into account the s_{err_i} contributions of all the operators in the layer. On the other hand, in the methodology proposed in this paper duplication is already applied at layer level, therefore the size of the solution space would remain the same, as well as the selective hardening strategy and related comments.

V. CONCLUSIONS

We presented a methodology aimed at improving the robustness of CNNs against SEU faults affecting the underlying hardware platform (a GPU). The strategy applies selective duplication of CNN layers driven by a Layer Vulnerability Factor metric to find a satisfactory robustness/performance trade-off. The proposal is based on a Design Space Exploration that explores the combinations of layer duplication estimating robustness and the execution times that allows the designer to compare alternatives. The size of the solution space is such that the DSE offers a viable systematic alternative to heuristically selected solutions, included in the result set.

REFERENCES

[1] M. Campbell, M. Egerstedt, J. How, and R. Murray, "Autonomous driving in urban environments: approaches, lessons and challenges," *Philosophical Trans. of the Royal Society A: Mathematical, Physical and Engineering Sciences*, vol. 368, no. 1928, pp. 4649–4672, 2010.

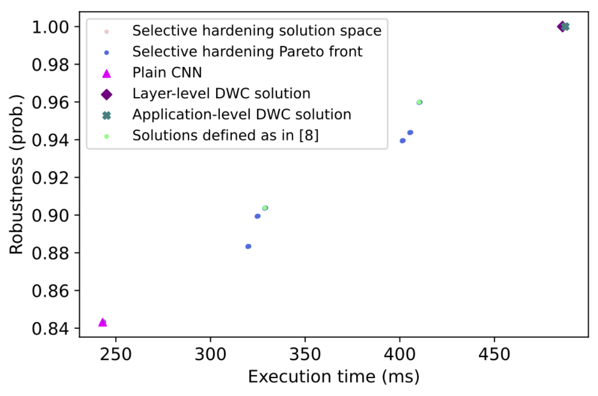
[2] R. Valiente, M. Zaman, S. Ozer, and Y. P. Fallah, "Controlling Steering Angle for Cooperative Self-driving Vehicles utilizing CNN and LSTM-based Deep Networks," in *Proc. Intelligent Vehicles Symp.*, 2019, pp. 2423–2428.

[3] Z. Ouyang, J. Niu, Y. Liu, and M. Guizani, "Deep CNN-Based Real-Time Traffic Light Detector for Self-Driving Vehicles," *IEEE Trans. Mobile Computing*, vol. 19, no. 2, pp. 300–313, 2020.

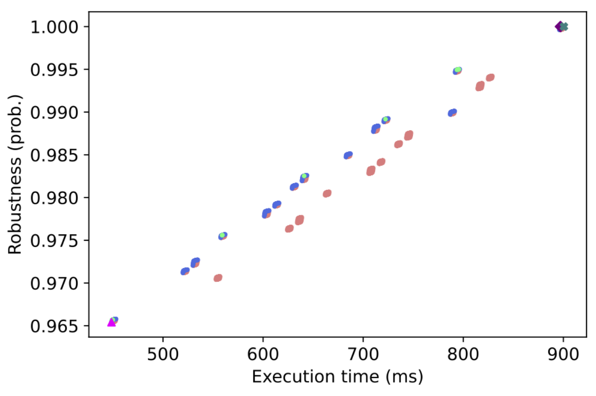
[4] International Organization for Standardization (ISO), "26262: Road vehicles - Functional safety," https://www.iso.org/standard/68383.html, 2011.

[5] T. Karnik and P. Hazucha, "Characterization of soft errors caused by single event upsets in CMOS processes," *IEEE Trans. Dependable and Secure Computing*, vol. 1, no. 2, pp. 128–143, 2004.

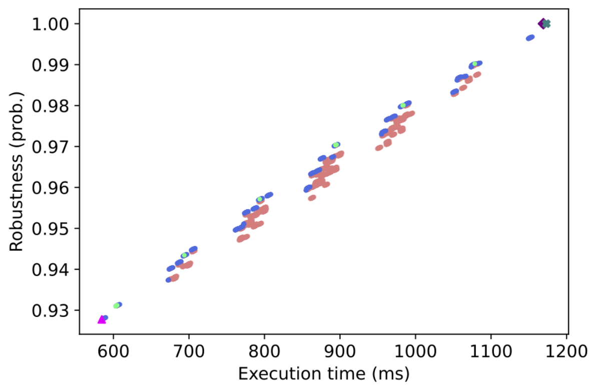
[6] E. Normand, "Single event upset at ground level," *IEEE Trans. Nuclear Science*, vol. 43, no. 6, pp. 2742–2750, 1996.



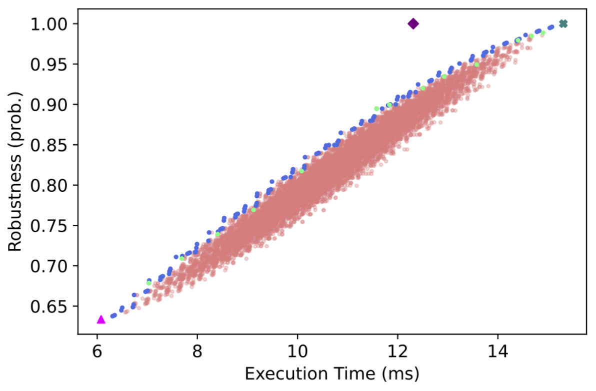
(a) Comma



(b) PilotNet



(c) Cifar10



(d) Vgg11

Figure 2. Design space exploration w.r.t. selective DWC-based hardening.

[7] ACEA - European Automobile Manufacturers' Association, "ACEA Report: Vehicles in use - Europe 2019," https://www.acea.be/publications/article/report-vehicles-in-use-europe-2019, 2019, (Accessed on 03/20/2020).

[8] F. F. dos Santos, L. Carro, and P. Rech, "Kernel and layer vulnerability factor to evaluate object detection reliability in GPUs," *IET Computers & Digital Techniques*, vol. 13, no. 3, pp. 178–186, 2019.

[9] F. F. dos Santos, P. Pimenta, C. Lunardi, L. Draghetti, L. Carro, D. Kaeli, and P. Rech, "Analyzing and Increasing the Reliability of Convolutional Neural Networks on GPUs," *IEEE Trans. Reliability*, vol. 68, no. 2, pp. 663–677, 2019.

[10] A. Mahmoud, S. K. S. Hari, C. W. Fletcher, S. V. Adve, C. Sakr, N. Shanbhag, P. Molchanov, M. B. Sullivan, T. Tsai, and S. W. Keckler, "Hardnn: Feature map vulnerability evaluation in cnns," *arXiv preprint arXiv:2002.09786*, 2020.

[11] A. Mahmoud, S. K. S. Hari, C. W. Fletcher, S. V. Adve, C. Sakr, N. R. Shanbhag, P. Molchanov, M. B. Sullivan, T. Tsai, and S. W. Keckler, "Optimizing selective protection for CNN resilience," in *Proc. Intl. Symp. Software Reliability Engineering*, 2021, pp. 127–138.

[12] M. Biasielli, C. Bolchini, L. Cassano, A. Mazzeo, and A. Miele, "Approximation-Based Fault Tolerance in Image Processing Applications," *IEEE Trans. on Emerging Topics in Computing*, vol. 10, no. 2, pp. 648–661, 2022.

[13] C. Bolchini, L. Cassano, A. Miele, and A. Toschi, "Fast and accurate error simulation for cnns against soft errors," *IEEE Trans. on Computers*, pp. 1–14, 2022.

[14] F. Libano, B. Wilson, J. Anderson, M. J. Wirthlin, C. Cazzaniga, C. Frost, and P. Rech, "Selective Hardening for Neural Networks in FPGAs," *IEEE Trans. Nuclear Science*, vol. 66, no. 1, pp. 216–222, 2019.

[15] B. Fang, K. Pattabiraman, M. Ripeanu, and S. Gurumurthi, "GPU-Qin: A methodology for evaluating the error resilience of GPGPU applications," in *Proc. Intl. Symp. Performance Analysis of Systems and Software*, 2014, pp. 221–230.

[16] G. Li, K. Pattabiraman, C.-Y. Cher, and P. Bose, "Understanding error propagation in GPGPU applications," in *Proc. Intl. Conf. High Performance Computing, Networking, Storage and Analysis*, 2016, pp. 240–251.

[17] T. Tsai, S. K. S. H. M. Sullivan, O. Villa, and S. W. Keckler, "NVBitFI: Dynamic Fault Injection for GPUs," in *Proc. Intl. Conf. Dependable Systems and Networks*, 2021, pp. 284–291.

[18] Z. Chen, N. Narayanan, B. Fang, G. Li, K. Pattabiraman, and N. DeBardeleben, "TensorFI: A Flexible Fault Injection Framework for TensorFlow Applications," in *Proc. Intl. Symp. on Software Reliability Engineering*, 2020, pp. 426–435.

[19] A. Mahmoud, N. Aggarwal, A. Nobbe, J. R. S. Vicarte, S. V. Adve, C. W. Fletcher, I. Frosio, and S. K. S. Hari, "PyTorchFI: A Runtime Perturbation Tool for DNNs," in *Proc. Intl. Conf. Dependable Systems and Networks Workshops*, 2020, pp. 25–31.

[20] M. A. Neggaz, I. Alouani, S. Niar, and F. Kurdahi, "Are CNNs Reliable Enough for Critical Applications? An Exploratory Study," *IEEE Design & Test*, vol. 37, no. 2, pp. 76–83, 2020.

[21] A. P. Arechiga and A. J. Michaels, "The Robustness of Modern Deep Learning Architectures against Single Event Upset Errors," in *Proc. High Performance extreme Computing Conf.*, 2018, pp. 1–6.

[22] A. Bosio, P. Bernardi, A. Ruospo, and E. Sanchez, "A Reliability Analysis of a Deep Neural Network," in *Proc. Latin American Test Symp.*, 2019, pp. 1–6.