ERODE: Error Resilient Object DetEction by Recovering Bounding Box and Class Information

ERODE: Error Resilient Object DetEction by Recovering Bounding Box and Class Information

Emanuele Valpreda*, Giuseppe Palumbo*†, Michele Caon*, Guido Masera*, Maurizio Martina*
*Department of Electronics and Telecommunications, Politecnico di Torino, Italy
†ST Microelectronics, Italy
Email:{emanuele.valpreda, michele.caon, guido.masera, maurizio.martina}@polito.it, giuseppe.palumbo@st.com

***Abstract*—Fault resilience in computer vision algorithms is paramount in critical applications such as autonomous driving or surveillance. Convolutional neural networks (CNNs) are usually used in these tasks to identify objects of interest, which are passed to other decisional algorithms and used to take specific actions. However, incorrect detections due to computation errors could pose a safety risk. In this work, we present ERODE (Error Resilient Object DetEction), a framework that can be paired with a CNN to filter the detections by identifying possible errors and restoring the correct predictions, improving the fault-resilience of the system. The proposed framework leverages the temporal correlation among consecutive images and CNN outputs, using motion estimation and tracking techniques to infer whether computation errors have occurred and, in that case, produce a new set of outputs. In order to evaluate the performance, precision and recall of the CNN with and without ERODE support have been computed and compared using the MOT17DET dataset, and EfficientDet D0 quantized to 16-bit, with errors injected in the activations computed during the inference. The experimental results show significantly reduced task accuracy degradation induced by bit-flips, proving that ERODE can increase the system's fault resilience.**

***Index Terms*—error detection, neural networks, object detection, object tracking, fault resilience**

I. Introduction

Convolutional neural networks (CNNs) have become the standard approach in computer vision tasks, such as image classification and object detection. They are increasingly used in safety-critical applications such as autonomous driving, or traffic surveillance [1], [2]. The presence of detection errors and glitches in the CNN output must be considered before the deployment to avoid safety risks. For example, the CNN could detect pedestrians, cars, or trucks in autonomous driving applications. The overall autonomous driving system will take decisions on accelerating, braking, or turning the car depending on the existence and position of those objects. A detection error can lead to an unsafe scenario for the driver, pedestrian, or objects surrounding the car. Atmospheric neurons, ionizing particles, voltage/temperature variations, and other interference may perturb a transistor's state, generating bit flips in memory or current spikes in logic circuits that, if latched, lead to an error [3], [4]. Moreover, bit flips in CNN's DRAM can be generated by row hammer attacks [5].

In this work, we present ERODE, a low-complexity approach to mitigate errors in the computation of CNNs for object detection, leveraging motion estimation techniques to predict the future position of known objects and eliminate temporary wrong detections. We use the spatio-temporal correlation between consecutive input images and output predictions to detect errors and eliminate glitches from the CNN output or restore the correct output that would have been corrupted otherwise. ERODE is orthogonal to any CNN model or hardware architecture executing it. It is developed to be a plug-in to preserve the task accuracy in the presence of errors. The remainder of this paper is organized as follows: Section II introduces relevant works addressing the topic of fault detection and resilience, Section III explains our methodology, Section IV presents the experimental results that validate our approach and Section V summarizes and concludes the paper.

II. Related Works

Draghetti et al. [3] propose to use inter-frame spatio-temporal correlation to detect errors in the CNN's inference. The basic assumption is that if the absolute pixel difference between two consecutive images is small, the inputs are almost equal, and the predictions made by the CNN will be very similar. If this condition does not occur, an error in the inference is hypothesized. Since the similarity is measured with a user-defined threshold, a change in brightness, noise, or camera movement could trigger a wrong error detection. In addition, the threshold would require fine-tuning for each environment in which the image sensor is used. In this work, we overcome this limit by using relative pixel differences, using well-known motion estimation techniques [6], [7], and considering the input-output spatio-temporal correlation over more consecutive images. Moreover, a deep analysis of faults occurring during the inference of quantized CNN models and their impact on the task accuracy is carried out in [4], [8]. These works highlight that activation errors impact CNNs' accuracy more than weight errors. Additionally, in [4], it is noted that modern CNNs are inherently more resilient to errors than older models, particularly when compressed through quantization for deployment. Therefore, in this work, we target bit-flips in activations and use EfficientDet D0, a modern CNN for object detection [9], quantized to 16 bits.

III. Method

ERODE leverages the spatio-temporal correlation between consecutive images within the video sequence to update the

position of detected objects using motion estimation techniques, adjusting the bounding box size and coordinates without processing the image with the CNN. Additionally, velocity and direction are evaluated to check that the trajectory of objects is constant or changes. This set of predictions and additional properties is compared against the predictions generated with the inference and is used to check the presence of errors by searching for abrupt changes in the identified objects' properties. For instance, a sudden change in the label attributed to a bounding box that had another one in the past images or an instant acceleration of a bounding box that was previously stationary could be caused by computation errors during the inference. The ERODE framework is depicted in Figure 1 and comprises of three parts, detailed later in this section: the tracker, the keep-alive register, and the predictor. The CNN generates the predictions, composed of bounding boxes and the corresponding label, passed to the tracker, which assigns a unique ID to each detected object. ERODE saves the IDs in the keep-alive register to identify the presence of objects through multiple images and selects which objects are relevant, i.e., persistent. The predictor then updates relevant object features (bounding box coordinates, dimensions, and velocity) by leveraging motion estimation techniques [6], [7]. The overall system's output comprises the keep-alive register entries, which substitutes the CNN's initial output predictions.

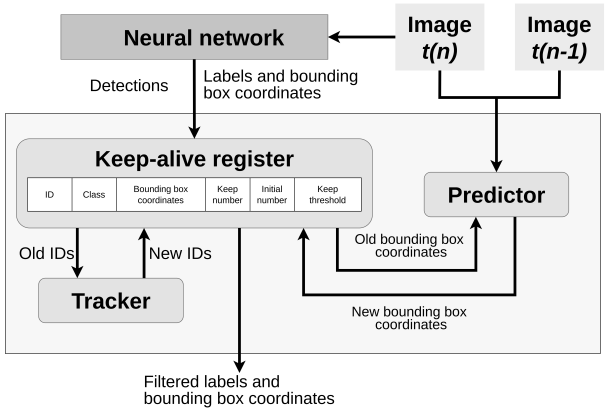

Fig. 1. The ERODE framework building blocks.

A. Tracker

The tracking algorithm gives a unique ID to each object in the image, which is used to observe its behavior in different time instants. In this work, we apply the tracking technique used in [10] to create a correlation between two consecutive CNN outputs and use the Hungarian algorithm [11] to solve the assignment problem, using the intersection over union (IoU) metric to compute the cost function. A cost matrix for each detection is generated from previously identified objects with labels and bounding boxes. First, the entries of the keep-alive register at the time $n-1$ (previous time-instant/image) are inserted in the rows, then the objects detected by the CNN at the time n (current image) are inserted in the columns. After that, the IoU between rows and column entries are evaluated. Where the IoU is high, the position in the current time instant is similar to the position in the past; therefore, objects of consecutive images can be associated with a unique ID encoded as a positive integer. On the other hand, objects with a low or zero IoU are considered new entries and are named with the first positive number available in the system. New entries could be either new relevant objects identified by the CNN or glitches due to computation errors. Finally, objects of the previous time instant with no association with those detected in the current one could no longer be present in the image or not detected due to computation errors.

B. keep-alive Register

The decision to assign, add or remove IDs is made considering the features related to every single object stored in the keep-alive register. When an object enters the image, it is immediately stored in the keep-alive register, depicted and described in Figure 2.

ID	Class	Bounding box coordinates	Keep number	Initial number	Keep threshold
1	Person	[20,87,43,45]	0	1	10
2	Car	[232,139,278,120]	2	1	8
⋮					
N	*xxxxxx*	$[x_l, y_t, x_r, y_b]$	*m*	1	*l*

Fig. 2. The keep-alive register is a data structure that contains features produced by the CNN (class, bounding box coordinates) and features produced by ERODE algorithms. The two objects in this figure are included to provide a hypothetical utilization.

For each CNN's detection, the keep-alive register is scanned to see if any objects are already stored in the structure. If the object is already in the register, that entry is updated with the newly detected features. If not, a new entry is created, and the first available number is assigned. The keep-alive register's primary function is to filter the CNN's detection. As stated previously, new detections could result from correct or faulty inference. In order to filter correct detections from errors, only objects that are present in multiple frames are shown in the system's output. The *initial number threshold* sets the minimum number of consecutive detections for each object, before they are considered correct and not glitches due to computation errors. This concept is borrowed from [12] and the *initial number threshold* indicates the degree of selectivity of the filter: a low threshold means that few detections in consecutive images are required to accept the object as an output of the system. The *initial number* counts the number of consecutive images from the first detection of the object, and, if it reaches the *initial number threshold*, the object can be considered as existing in the image and so it can be shown as an output of this system. On the contrary, the object is deleted from the system's memory because it is identified as an error. Any object in the keep-alive register can have two different fates when the subsequent detection is performed: if detected, the algorithm assumes that it is present in the image, and so that the detection is correct, whereas if it is not detected, it could be due to an error or to the object actually leaving the

image. To model this, keep-alive features are used. Similarly, objects which are not detected for one or two frames could no longer be present in the image or again be the result of computation errors. The *keep number* indicates the number of consecutive images in which the CNN does not detect an object. If the object is detected, this number is reset to zero for all its consecutive detections. Therefore, the *keep number* gives essential information to the overall algorithm, indicating when a prediction is needed. If the *keep number* is equal to zero, the object's position is given by the detector; otherwise, it must be evaluated using the predictor. Since the predicted position is a guess and the object can leave the image anytime, a limitation on the number of consecutive times a prediction can be used is given [13]. This limit is set by the *keep threshold*, a user-defined parameter and is increased for each consecutive detection and, conversely, decreased for each non-detection.

C. Predictor

The predictor is used when an object is undetected for one or more consecutive frames, i.e., the *keep number* is greater than zero. The predictor updates the position of known objects by searching for similarities between the current and previous image, using points of interest derived as the center of the objects' bounding boxes. The predictor uses the diamond search algorithm proposed in [7], with a search window of 15, to estimate the movement and new coordinates of the bounding boxes of relevant objects. This motion estimation algorithm was selected due to its low computational complexity and adaptability to different ranges of motions.

IV. Results

This section details the computing setup and results, which comprises the neural network, the error injection algorithm, the ERODE framework, and the benchmarks. In this work, the detector used to generate the predictions is EfficientDet D0 [9], a modern CNN architecture for object detection, with activations and weights quantized to 16 bits using scale quantization [14]. The model is then retrained for 20 epochs using the hyperparameters detailed in [9] to recover from the accuracy loss induced by the quantization. The fault-free CNN is then used to generate the baseline detections set for each image of each sequence of MOT17DET [15], a dataset of street scenes with images captured with stationary and moving cameras. The baseline detection set is used to evaluate the accuracy loss of the CNN executed with computation errors with and without the support of ERODE, noted as faulty CNN and ERODE, respectively in Figures 3 and 4. To evaluate the error recovery capabilities of ERODE, we consider a scenario where errors occur not on each image but with a frequency defined as image error rate Δ. For the experiments presented in this section, we set $\Delta \in [2, 3, 4, 5]$, which allows testing the effectiveness of the keep-alive register in filtering wrong detections occurring at different rate. For instance, a Δ equal to two means that one every two images will be processed with faults injected during the inference. Moreover, we define the activations error rate ε of each frame as the percentage of the total computation volume subjected to bit flips. In the experiments we set $\varepsilon \in [0.001\%, 0.003\%, 0.005\%]$. The selection of which activations are subjected to bit-flips and which bits are flipped is made by sampling random integer numbers from a uniform distribution, meaning that each activation within the tensor and each bit within the activation have an equal probability of being selected. Only one bit can be flipped for each activation. The number of errors generated for each combination of Δ and ε is reported in Table I.

TABLE I
ERRORS INJECTED DURING THE INFERENCE

	Δ = 2	Δ = 3	Δ = 4	Δ = 5
ε = 0.001%	11538852	7679515	5765075	4599007
ε = 0.003%	35107176	23365070	17540350	13992566
ε = 0.005%	58712628	39075335	29334175	23400923

Faults are distributed uniformly over each convolutional layer of EfficientDet D0 under the assumption that computation errors occur during the entire inference process.
To evaluate the effectiveness of the proposed method, we consider true positives detections with an IoU that is at least 50% with the baseline, false positives detections that are not present in the baseline, and false negatives detections that are only present in the baseline. Precision and recal are evaluated as in Equation 1.

$$Precision = \frac{TP}{TP + FP} \qquad Recall = \frac{TP}{TP + FN} \tag{1}$$

The keep-alive register, presented in Section III-B, has three parameters that must be defined by the user. Recalling that the *initial filter threshold* sets the consecutive detections required to consider an object as correctly identified, the *keep number threshold* sets the limit on consecutive non-detections before the predictor is used to evaluate the position of the object, and the *keep-alive threshold* sets the limit on non-detections before the object is removed from the register, we set: *keep-alive threshold*=10, *keep number threshold*=5, *initial filter threshold*=1. The precision variation with different Δ and ε is depicted in Figure 3. First, it is possible to notice how EfficientDet D0 is very susceptible to computation errors, as a ε=0.003% is high enough to induce task accuracy loss of 40% compared to the baseline for Δ=2 and ε=0.005% further drops the accuracy to 70%. When Δ is increased, inference errors occur less frequently, and the overall precision loss is reduced. Similarly, the same behavior can be seen for the recall results in Figure 4. The discrepancy between the precision and accuracy loss could mean that the CNN tends to see non-existing objects rather than not detecting existing ones. An inspection of a subset of output predictions highlighted the presence of multiple wrong objects, with a small variation of the bounding box coordinates of correct objects. When the CNN is supported with ERODE, the accuracy and recall loss is significantly reduced. By leveraging the keep-alive register, it is possible to filter false positives, using the *initial filter* to eliminate detections that occur only in the Δ images due to computation errors. Moreover, missed detections are restored

with the predictor after the object is not detected for a number of frames higher than the *keep number threshold*. ERODE's precision degradation could be due to the predictor's bounding box estimation, which might not overlap precisely with the ones generated with the baseline CNN, and to objects no longer present in the frame that persists for a few time instants in the keep-alive register. Moreover, the initial filter could also remove correct objects not detected continuously in all images, negatively affecting the recall by increasing false negatives. This can be mitigated by improving the accuracy of the CNN, so that it detects objects more consistently or by decreasing the *initial number threshold*.

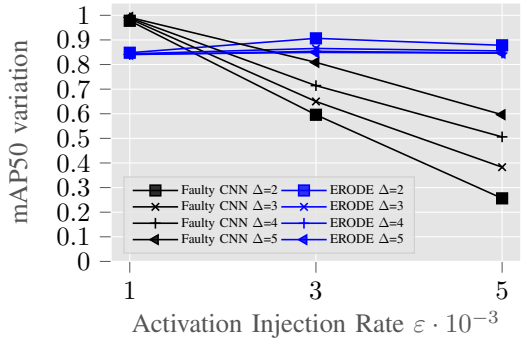

Fig. 3. Precision variation with different Δ and ε

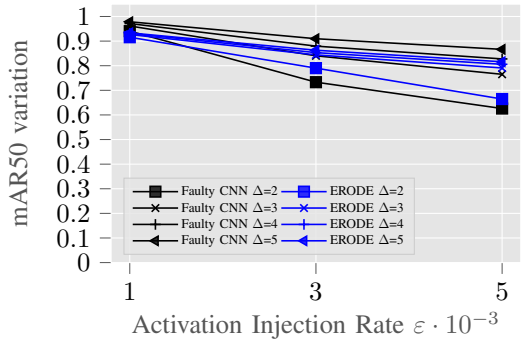

Fig. 4. Recall variation with different Δ and ε

Given the above, ERODE can be effectively used for error-resilient object detection, as it can filter out wrong detections and recover information lost due to computation faults. Finally, we estimate the computational complexity as the average number of multiplications and sums required to execute ERODE for each image, reported in Table II. The complexity increases with the error rate as more glitches are generated in the CNN's output, resulting in more bounding boxes evaluated by ERODE. The overhead is, therefore, negligible compared to the number of multiply and accumulate operations required to compute EfficientDet D0, which is ≈2.5 billion [9].

V. Conclusion

In safety-critical applications, it is important to ensure fault-resilient execution of CNN models. In this work, we presented ERODE, a lightweight framework that can detect errors and recover lost objects' information, i.e., class and coordinates, reducing the task accuracy drop and effectively increasing the fault resilience of the system. We detailed the blocks composing the system and explained the influence over the system's output. Finally, we introduced the experimental setup used to validate ERODE and presented the results that prove the effectiveness of our approach.

TABLE II
Computational complexity of the motion estimation algorithm averaged over all the images of MOT17DET.

		$\Delta = 2$	$\Delta = 3$	$\Delta = 4$	$\Delta = 5$
$\varepsilon = 0.001\%$	Products	48	49	49	49
	Sums	2565	2599	2611	2619
$\varepsilon = 0.003\%$	Products	34	43	47	48
	Sums	3394	3064	3519	3267
$\varepsilon = 0.005\%$	Products	97	82	79	64
	Sums	14963	9166	8984	5915

Acknowledgment

We thank the Nvidia's Academic Grant for donating the Quadro RTX A5000 GPU used in this work.

References

[1] F. Zhang *et al.*, "Cmnet: A connect-and-merge convolutional neural network for fast vehicle detection in urban traffic surveillance," *IEEE Access*, 2019.

[2] D. B *et al.*, "Improved object detection in video surveillance using deep convolutional neural network learning," *I-SMAC*, 2021.

[3] L. K. Draghetti *et al.*, "Detecting errors in convolutional neural networks using inter frame spatio-temporal correlation," *IOLTS*, 2019.

[4] N. Fasfous *et al.*, "Mind the scaling factors: Resilience analysis of quantized adversarially robust cnns," in *DATE*, 2022.

[5] M. Son *et al.*, "Making dram stronger against row hammering," in *DAC*, 2017.

[6] W. Hassen and H. Amiri, "Block matching algorithms for motion estimation," *ICELIE*, 2013.

[7] S. Zhu and K.-K. Ma, "A new diamond search algorithm for fast block-matching motion estimation," *IEEE Transactions on Image Processing*, vol. 9, no. 2, 2000.

[8] Y. He *et al.*, "Fidelity: Efficient resilience analysis framework for deep learning accelerators," in *MICRO*, 2020.

[9] M. Tan, R. Pang, and Q. V. Le, "Efficientdet: Scalable and efficient object detection," in *CVPR*, 2020.

[10] A. Bewley *et al.*, "Simple online and real-time tracking," *ICIP*, 2016.

[11] H. W. Kuhn, "The hungarian method for the assignment problem," *Naval Research Logistics Quarterly*, 1955.

[12] E. Bochinski *et al.*, "High-speed tracking-by-detection without using image information," *AVSS*, 2017.

[13] E. Bochinski *et al.*, "Extending iou based multi-object tracking by visual information," *AVSS*, 2018.

[14] H. Wu *et al.*, "Integer quantization for deep learning inference: Principles and empirical evaluation," *arXiv preprint arXiv:2004.09602*, 2020.

[15] A. Milan *et al.*, "MOT16: A benchmark for multi-object tracking," *arXiv:1603.00831 [cs]*, Mar. 2016.