Fault Impact Estimation for Lightweight Fault Detection in Image Filtering

Cristiana Bolchini, *Senior Member, IEEE,* Giacomo Boracchi, *Member, IEEE,*
Luca Cassano, *Member, IEEE,* Antonio Miele, *Senior Member, IEEE,* Diego Stucchi

Abstract—Classical redundancy-based fault detection techniques, such as Duplication with Comparison (DWC), rely on replicating the computation and comparing the replicas' output at a bit-wise granularity. In many application environments these costs are prohibitive, especially when applications are characterized by an intrinsic level of tolerance. This paper presents a novel fault-detection approach for the specific context of image filtering. Peculiarity of the proposed approach is that it estimates the impact of the fault on the processed output, in order to determine whether the image is usable or should be re-processed. To limit overheads, the proposed solution exploits Approximate Computing (AC), allowing the definition of disciplined AC strategies to trade-off between accuracy and costs. Core of our solution is the successful combination of Image Quality Assessment metrics and Machine Learning models to assess the visual impact of the fault in a lightweight manner. Extensive experimental campaigns demonstrate the effectiveness of the solution, achieving achieving a reduction in terms of execution time up to 44% with respect to the classical DWC, with a fault detection precision ranging from 94.58% to 96.70%, and recall ranging from 88.2% to 97.8%, depending on the adopted level of approximation.

Index Terms—Approximate Computing, Fault Detection, Image Processing, Neural Networks, Image Quality Assessment Metrics.

1 Introduction

In several classes of systems, safety-/mission-critical applications and non-critical ones coexist. In the aerospace domain, for example, satellites are equipped with several mission-critical modules (e.g. Attitude Determination and Control Subsystem), and other non-critical payload processing applications [1]. While fault tolerance is imperative for mission-critical components, less stringent and less expensive solutions can be adopted to monitor payload applications, and also to limit the processing overheads due to faults that do not significantly impact the processed data.

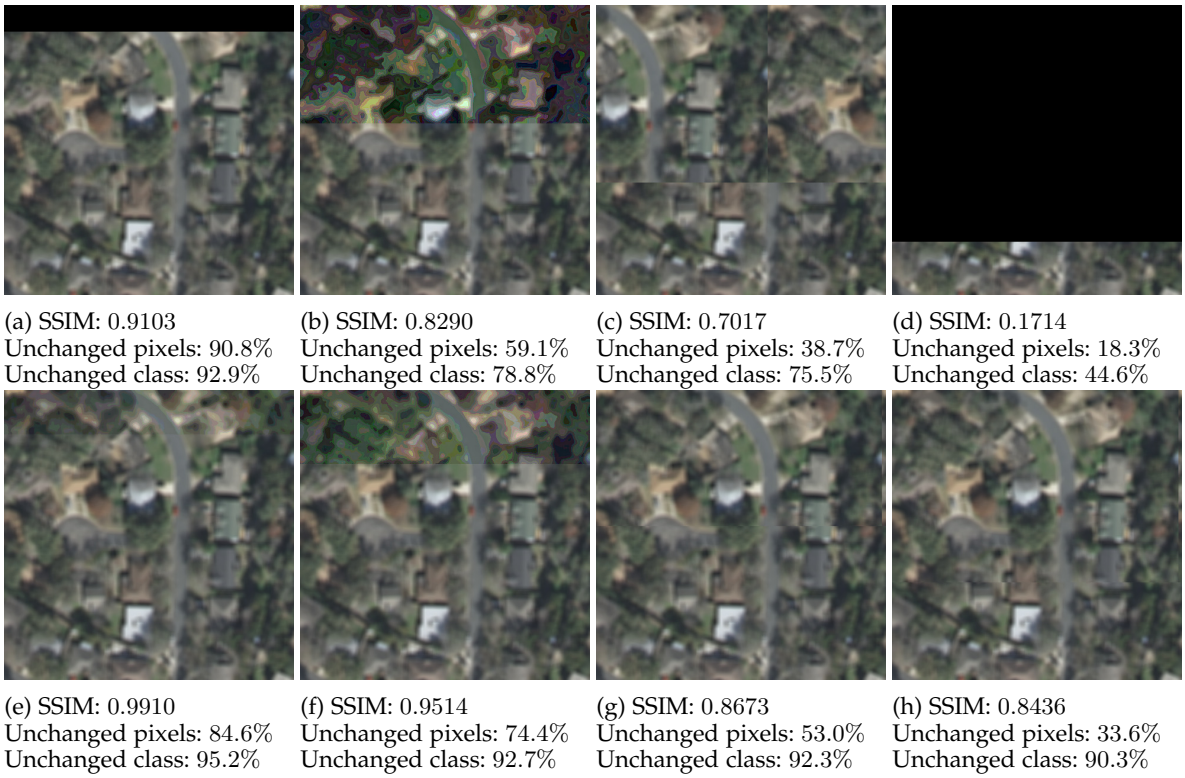
For example, payload applications implementing image processing functionalities, such as filters, are often characterized by an intrinsic level of tolerance, as these deal with noisy inputs and directly provide output images to humans or to other software performing probabilistic estimates (e.g. classifiers). In this scenario, fault-detection schemes based on Duplication with Comparison (DWC) may be too stringent, since the Two-Rail Checker (TRC) would discard the two processed images when these differ, even in a single pixel. This scheme is therefore highly inefficient since it discards *slightly* corrupted images, which would be effectively used by the payload application or the user, and activates unnecessary processing overheads. Recently, a fault management scheme based on image *usability* has been proposed [2] to mitigate these problems.

We present the Fault Impact Estimator (FIE), a novel solution to detect faults in image processing applications. The key idea behind the FIE is to adopt an Image Quality Assessment (IQA) metric – such as Structural Similarity index (SSIM index) [3] – to quantitatively assess the *visual impact* of a fault occurred while processing the input image. The following example illustrates that an IQA well correlates with the severity of the fault, and that in some cases this can discriminate better than classical solutions whether the output image would be usable or not.

Fig. 1 shows eight faulty outputs of a smoothing filter, which attenuates the high frequencies of an image. The caption of each image reports *i)* the SSIM index value as a way to quantitatively assess the visual difference between the fault-free and the faulty output (no difference corresponding to 1 and large differences approaching 0), *ii)* the percentage of pixels where faulty and non-faulty outputs differ, that is what DWC would report, and *iii*) the percentage of pixels where the output of a simple payload application has changed. First of all, Fig. 1a-d indicate that the SSIM index is a meaningful metric to assess the impact of the fault, as it correctly decreases when the fault becomes more impactful. The second row reports a situation where the SSIM index can assess the fault impact much better than the percentage of altered pixels, thus it can determine whether the output would be usable or not. In fact, faults in Fig. 1e-h do not exhibit relevant differences, and all provide a low impact in terms of application. Here, the SSIM index turns to be a meaningful indicator of the visual impact of these faults, while counting the number of altered pixels is useless to determine whether the image is usable or not, as shown by the large variability of the unchanged pixels percentages.

The proposed FIE is an innovative fault management strategy, to be used at the application-level, which has been specifically tailored for image processing applications. In particular, the FIE has been designed to be lightweight, requiring much less operations than the classical DWC and other detection schemes based on replication. The FIE relies on two major ingredients:

- *Authors are with the Dipartimento di Elettronica, Informazione e Bioingegneria, Politecnico di Milano, Italy.*
E-mail: {first_name.last_name}@polimi.it



(a) SSIM: 0.9103
Unchanged pixels: 90.8%
Unchanged class: 92.9%

(b) SSIM: 0.8290
Unchanged pixels: 59.1%
Unchanged class: 78.8%

(c) SSIM: 0.7017
Unchanged pixels: 38.7%
Unchanged class: 75.5%

(d) SSIM: 0.1714
Unchanged pixels: 18.3%
Unchanged class: 44.6%

(e) SSIM: 0.9910
Unchanged pixels: 84.6%
Unchanged class: 95.2%

(f) SSIM: 0.9514
Unchanged pixels: 74.4%
Unchanged class: 92.7%

(g) SSIM: 0.8673
Unchanged pixels: 53.0%
Unchanged class: 92.3%

(h) SSIM: 0.8436
Unchanged pixels: 33.6%
Unchanged class: 90.3%

Figure 1: Faulty outputs of the convolution of an input image I with a smoothing filter f. *(a-d)* An IQA metric (SSIM index) provides a reliable estimate of the severity of the fault *(e-h)* and it can estimate the fault better than the pixel-wise difference count.

- Approximate Computing (AC) that defines an *approximated replica* instead of replicating the nominal processing to reduce the overheads.
- A regression model from the Machine Learning (ML) literature that estimates the value of an IQA metric measuring the fault impact.

On the one hand, replication based on AC dramatically reduces the computational cost with respect to the classical DWC scheme, where replication introduces more than 100% overheads in terms of area/time. On the other hand, since the nominal output and approximated replica are typically different even in fault-free situations, replication based on AC prevents to directly compute the IQA metric. To this purpose, we provide a methodology for adopting a ML model for regression that can estimate the IQA value by comparing the nominal output and the approximated replica.

The proposed approach is very general and can be adopted for many types of image processing operations, IQA metrics, and ML models. Here we consider 2D convolution as target application, as this describes any linear and space invariant operation on images, which are ubiquitous in image processing pipelines. Nonetheless, we show that the FIE can be used even in the extreme case where the approximated replica coincides with the input (thus reducing to zero the replication cost), indicating that this is a viable approach for other classes of filters and processing blocks, disregarding the type of approximation.

Moreover, the FIE features some very practical advantages which can ease its development in real-world scenarios. In particular, since the regression model is trained to estimate an IQA metric, the whole system can be set-up in a completely unsupervised manner, without requiring either experts or oracles to annotate images as usable/unusable, as in [2]. The FIE can be also easily adapted when the operating conditions change (e.g. when the spatial resolution of the imaging apparatus varies or the employed filter changes) without requiring a re-design of the solution from scratch.

Experiments, performed on large datasets of aerial images, show that the FIE i) provides a flexible framework to accurately estimate the fault impact at various levels of AC, ii) can be used to determine whether the output of an image processing pipeline is usable or not, and iii) favourably compares against the classical DWC solution in terms of execution times. This is particularly evident when we push AC to the limit, and estimate the fault impact by comparing the input image with the faulty output.

Our paper is organized as follows. Section 2 introduces a few basic notions. The architecture of FIE is presented in Section 3, while design choices and steps are discussed in Section 4. Extensions over the basic FIE scheme are presented in Section 5. Results of our extensive experimental campaign are reported in Section 6. Section 7 overviews the state-of-the-art solutions discussing limitations and the gaps our solution aims at filling, and finally Section 8 concludes the paper.

2 Preliminaries

Here we present the preliminary information to contextualize our proposal, setting the background and the baseline alternatives.

2.1 Fault Model and Management

Our goal is to design a fault management scheme that supports reliability against Single Event Upsets (SEUs) in computing systems running image processing applications. SEUs may cause different effects; i) system crashes, ii) non-terminations, and iii) Silent Data Corruptions (SDCs). While faults belonging to the first two classes are easily detectable since the system stops responding, faults causing SDCs are the most critical ones because the system returns an incorrect result with no additional evidence of the occurred anomaly. Our aim is to provide protection for image processing tasks against SDCs, to trigger a re-computation only when the fault has a large impact and the output image would not be considered *usable*. To this end, the FIE significantly reduces the average execution times with respect to traditional approaches. We assume at most one fault may occur during a single run of the application.

2.2 Considered Class of Filters

We focus on the large class of convolutional filters to present our methodology. These represent the building blocks of many image processing pipelines and algorithms, ranging from noise suppression [4] and contrast enhancement – which are typically performed as preprocessing steps – to feature extraction, which underpins high-level visual recognition algorithms [5]. Each of these filters is identified by a 2D matrix f, typically having much smaller size than the input image I, which defines the convolution output

$$(I \circledast f)(r,c) = \sum_{i,j=-H,\dots,H} f(i,j)I(r-i,c-j) \qquad (1)$$

where $\circledast$ is the convolution symbol, (r,c) are the coordinates in the filtered image and the filter size is $L = 2H+1$, assuming the filter to be square for the sake of simplicity.

We test our solution on both smoothing filters, widely used in denoising and to suppress high-frequency components, and Laplacian filters, used to improve contrast and enhance edges and details.

2.3 Image Quality Assessment Metrics

The visual quality of a digital image is affected by a wide variety of image distortions, thus Image Quality Assessment has attracted a lot of interest [6] [7]. In particular, most of these IQA metrics to be computed require a reference image free of distortions. We propose to exploit IQA metrics to assess the fault impact, where the distorted image corresponds to the possibly faulty output of an image processing pipeline. Our FIE can be used to predict any IQA metric it is trained for, and in our experiments we consider three different metrics: the Root Mean Square Error (RMSE), the SSIM index [3] and the Gradient Similarity Metric (GSM) [8]. While IQA metrics are typically meant for processing natural images, we use them on filtered outputs. Thus, the IQA has to be wisely selected to provide meaningful assessment of the fault impact.

2.4 Baseline Solutions

We will compare the performance of the FIE with two already mentioned alternative schemes. The first one consists in processing the input image twice, to obtain an exact replica and then counting the number of pixels where the two outputs differ. Images where the number of different pixels exceed a threshold are discarded. We refer to this solution as *flexible DWC*, as this is a relaxation of the classical DWC with TRC scheme. The second solution is denoted as *Dupl+IQA* and computes an exact replica, followed by the computation of an IQA metric to measure the fault impact. The value of this metric is then used to determine if the output is usable or not.

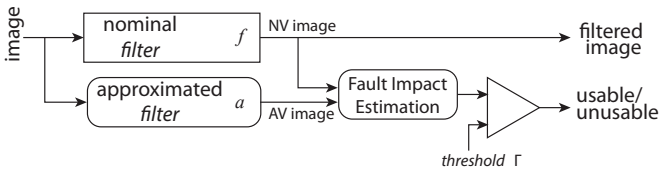


Figure 2: Our fault impact estimation strategy.

3 The Fault Impact Estimator

This section presents the FIE architecture, our major contribution. We first describe how to use this module in a fault-detection scheme, then we provide a formal description of the fault impact estimation problem, and finally describe the FIE in details.

3.1 Fault Detection Scheme

Fig. 2 illustrates the fault-detection scheme where the FIE can be employed. As in DWC, each input image is processed twice, but instead of two exact replicas, there are a nominal processing and an approximate one. In the considered case study, these are the convolution against a nominal and an approximated filter, f and a, respectively. The two outputs are then fed to the FIE module that assesses the visual impact of the fault (if any) by estimating $\widehat{m}$, namely the value of the selected IQA metric. The estimated fault impact is then compared against a threshold Γ to determine whether it is sufficiently large and the output should be discarded as unusable. Our strategy is robust and general, since the design and implementation of the FIE does not depend on the IQA metric threshold Γ which can be set by the designer for the specific application context, as discussed later in Section 4. Moreover, the level of approximation in the FIE can be tuned depending on the available resources, and the FIE has to be designed and trained accordingly.

3.2 The Fault Impact Estimation Problem

To ease the description of the FIE, we first define the problem of estimating the impact of a fault $\mathcal{G}$ corrupting an image filtering module. Let us consider the 2D convolution of an input image $I \in \mathbb{R}^{R\times C}$ against a filter $f \in \mathbb{R}^{L\times L}$, which we refer to as the *nominal* filter[1]. We indicate by $I \circledast f$ the output of the convolution when no fault occurs, and by $\mathcal{G}(I \circledast f)$ the output corrupted by a fault $\mathcal{G}$.

1. In color images the convolution is typically being applied on each color plane separately, thus we consider 2D convolutions only.

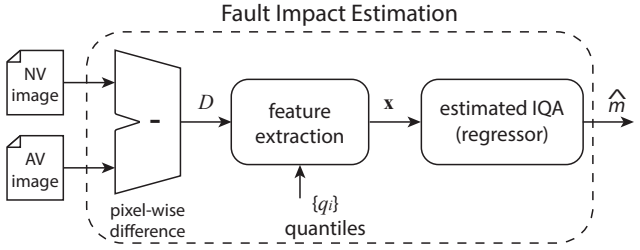


Figure 3: The proposed FIE module.

Our goal is to estimate the fault impact, which on the convolution operation can be defined as:

$$m = \mathcal{M}\left((I \circledast f), \mathcal{G}(I \circledast f)\right), \tag{2}$$

where $\mathcal{M}$ is the IQA metric selected to quantitatively assess the effects of the fault $\mathcal{G}$ with respect to the fault-free output $(I \circledast f)$. As illustrated in Section 1 (Fig. 1), this IQA metric can be used determine whether the output $\mathcal{G}(I \circledast f)$ is usable or not and the FIE module efficiently estimates the fault impact as in Eq. (2) without the need of replicating $(I \circledast f)$.

3.3 FIE Scheme

Fig. 3 illustrates the scheme of the proposed FIE which estimates the fault impact Eq. (2) from the nominal output $(I \circledast f)$ and its *approximated replica* $(I \circledast a)$ by:

1) computing the pixel-wise difference D between the nominal output and approximated replica;
2) extracting a feature vector $\mathbf{x} = \mathcal{E}(D) \in \mathbb{R}^d$, that provides a compact description of the values in D;
3) estimating the fault impact $\widehat{m} = R(x)$ by means of a regression model R that takes as input a feature vector $\mathbf{x}$.

In formulas, the FIE computes an estimate $\widehat{m}$ of the fault impact m in Eq. (2) as follows:

$$\widehat{m} = R(\mathcal{E}(\mathcal{G}(I \circledast f) - (I \circledast a))), \tag{3}$$

where $\mathcal{E}$ denotes the operator extracting the feature vector from the nominal and approximated replica. In the following we describe the modules corresponding to these operations discussing the rationale underpinning our design choices.

Computing Output Difference

We compute the difference image $D \in \mathbb{R}^{R \times C}$ as the pixel-wise difference between the nominal output and approximated replica, namely:

$$D(r,c) = (I \circledast f)(r,c) - (I \circledast a)(r,c) \tag{4}$$

Due to approximation, this is typically nonzero even in fault-free conditions, as shown in Fig. 4.

The assumption underpinning FIE follows from this observation: since convolution is a linear operation, D can be seen as the convolution of I against the filter $(f - a)$. Then, when I belongs to a specific type of images (e.g., aerial images at a fixed resolution), for each pair of filters f and a and in absence of faults, the values in D follow an unknown but fixed distribution. In contrast, when the fault has a severe impact, values in D follow a different distribution.

Histograms in Fig. 5 illustrates this fundamental principle.

Feature Extraction

The distribution of values in D is difficult to handle, even in the form of a histogram, as it is very high-dimensional. For this reason, we extract a feature vector $\mathbf{x} \in \mathbb{R}^d$ as a compact and efficient-to-compute descriptor of the shape of the distribution of D. Each component of $\mathbf{x}$ is defined as

$$x_i = \frac{\#\{(r,c), D(r,c) > q_i\}}{R \cdot C}, \quad i = 1, \ldots, d \tag{5}$$

where $\#$ denotes the cardinality of a set and each component x_i of $\mathbf{x} \in \mathbb{R}^d$, contains the proportions of pixels in D exceeding a fixed quantile q_i. Thus, extracting a feature vector corresponds to sampling the empirical cumulative density function of D at pre-defined cut-points (quantiles) $Q = \{q_i\}_{i=1}^d$, see Fig. 5b. Features are extracted by the operator $\mathcal{E}(\cdot)$, which returns a feature vector from an input image I, namely $\mathbf{x} = \mathcal{E}_{f,a,Q}(I)$. The operator $\mathcal{E}$ obviously depends on the filters f, a, and on the quantiles Q.

Among other options for describing a distribution, we select Eq. (5) because of its efficiency, since the feature vector $\mathbf{x}$ can be computed from a single pass over D and at most d comparisons against fixed quantiles Q. From the example in Fig. 5a, we see that faults affect the distribution of the values in D. In particular, we notice that faults having a high-impact on the image, also have a high impact on this distribution. This is even more evident when comparing the empirical cumulative distributions shown in Fig. 5b, hence when comparing the respective feature vectors.

The set of quantiles Q plays a central role in the FIE and these quantiles are strongly dependent on the class of images $\mathcal{I}$ being processed and on the pair of filters (f, a). In Section 4, we describe how these can be obtained from a set of images without faults, while in Section 5.1 we show that adapting these quantiles provides a lot of flexibility to the FIE in many practical circumstances.

A Machine Learning model for IQA Metric Estimation

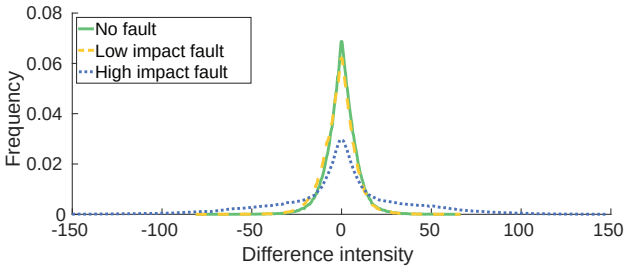
The last step performed by the FIE is to estimate the fault impact. A feature vector might reveal when a fault is heavily corrupting the output image (see Fig. 5), but there is no analytical expression leading from $\mathbf{x}$ to the fault impact in Eq. (2). Therefore, we resort to an ML model and train a regressor such as an Linear Model (LM) or an Neural Network (NN) that takes as input $\mathbf{x}$ and returns an estimate of the corresponding fault impact for a specific quality metric $\mathcal{M}$.

4 The FIE Design

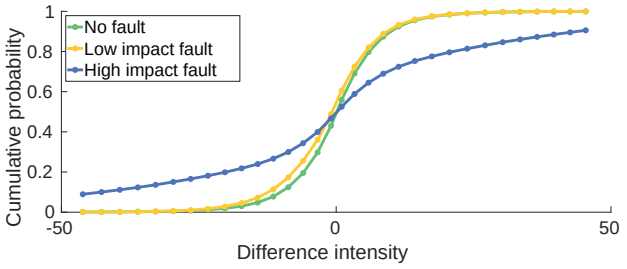
A design methodology has been defined for both configuring and training the FIE, and to support the proposed fault detection strategy. Fig. 6 illustrates the design flow, which depends on the filters f and a, on the dataset $\mathcal{I}$ containing images from the considered application (e.g., aerial images), and on the target processing platform with the adopted fault model. This is composed of the following key steps.

(a) Input image I (b) Nominal output $(I \circledast f)$ (c) Approximate output $(I \circledast a)$ (d) Detail of the nominal output (e) Detail of the approximate output

Figure 4: Input image I and LoG nominal and approximate outputs, with details.



(a) Density distributions of D



(b) Cumulative distributions of D

Figure 5: Density and cumulative distributions of the values in D at different level of fault impact (images in Fig. 1e and Fig. 1c).

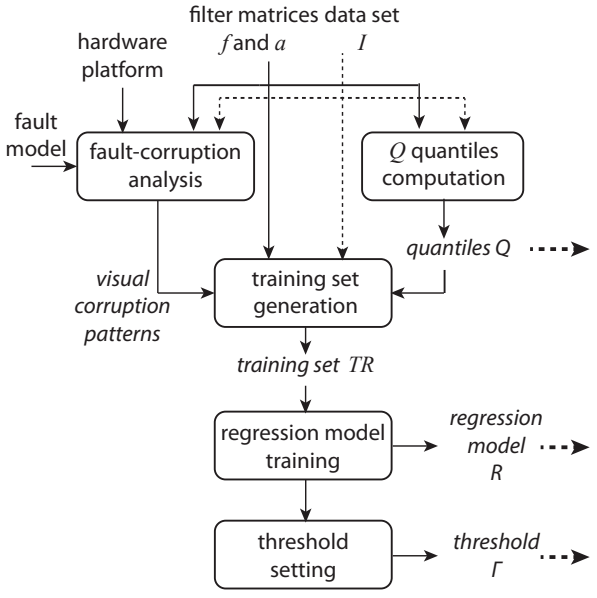


Figure 6: The FIE design methodology.

Quantiles Computation

To extract features we define cut-points $Q = \{q_i\}_{i=1}^{d}$ where the d quantile represents the empirical distribution of pixel-wise differences in nominal-approximate pairs. Therefore, we randomly select a few dozens of images from $\mathcal{I}$, compute for each one of these D as in (4), and accumulate all the values in a histogram to compute the quantiles Q. Since we expect faults to mainly affect the tails of this distribution, we select symmetric quantiles that are denser close to 0% and 100%, and sparser around 50%. We also insert six fixed thresholds $\{w_j\} = \{-250, -200, -150, 150, 200, 250\}$ to take into account large pixel differences that rarely occur in fault-free conditions (thus would not be estimated from a few images). We experienced that the latter can significantly improve the accuracy of the FIE. In the following, unless differently stated, the *quantiles set* is $Q = \{q_i\} \cup \{w_j\}$.

Fault Corruption Analysis

We here adopt a strategy similar to the one proposed in [2], [9] to generate a large set of corrupted images corresponding to the adopted fault model (e.g. SEUs), target platform and image processing application being executed. We combine the accuracy of fault injection with the controllability of error simulation in a two-step flow. First, architecture-level fault injection experiments are performed on the real hardware platform while executing the application to collect a number of corrupted images. Then, the collected images are analyzed to extract a number of *visual corruption patterns* representative of the effects that hardware fault may induce in the produced output images. Such visual corruption patterns are finally integrated into a simulator to generate the training set for the regression model, as described in the following.

Training Set Generation

To train the regression model we generate a large training set TR containing pairs of feature vectors $\mathbf{x}$ and corresponding quality metric m:

$$TR(\mathcal{I}, f, a, Q) = \{(\mathbf{x}, m) | \mathbf{x} = \mathcal{E}_{f,a,Q}(I), I \in \mathcal{I}\} \qquad (6)$$

where m is computed as in Eq. (2).

As described in Algorithm 1, for each image we first compute the feature vectors in a fault-free situation (Lines 3–4). Since in this case $\mathcal{G}(I \circledast f) = (I \circledast f)$, we associate with these feature vectors the best quality metric value $m = B_{\mathcal{M}}$ (eg. RMSE=0, SSIM=1), as stated by Eq. (2). For TR to

Algorithm 1 The training set generation procedure

Inputs:
$\mathcal{I}$: data set of images
f, a: nominal and approximate filter
$\mathcal{M}, B_{\mathcal{M}}$: quality metric and its maximum value.
E: list of visual corruption patterns
Outputs:
TR: training set containing pairs of feature vectors ($\mathbf{x}$) and the corresponding quality metric values (q).

Body:

1: $TR \leftarrow \{\}$
2: **for all** $I \in \mathcal{I}$ **do**
3: $\mathbf{x} \leftarrow$ extract_features($(I \circledast f)$, $(I \circledast a)$)
4: $TR \leftarrow (TR \cup \{(\mathbf{x}, B_{\mathcal{M}})\})$
5: **for** $e \leftarrow 1$ **to** E **do**
6: $\mathcal{G} \leftarrow$ select_random_corruption_pattern(e)
7: $\mathcal{G}(I \circledast f) \leftarrow$ inject_corruption_pattern($(I \circledast f)$, $\mathcal{G}$)
8: $\mathbf{x} \leftarrow$ extract_features($\mathcal{G}(I \circledast f)$, $(I \circledast a)$)
9: $m \leftarrow$ compute $\mathcal{M}\left(\mathcal{G}(I \circledast f), (I \circledast f)\right)$
10: $TR \leftarrow (TR \cup \{(\mathbf{x}, m)\})$

be representative of all the possible effects of faults, thus allowing the FIE module to learn all of them, we perform multiple executions of the application on every image in the dataset, injecting one of the previously visually identified corruption models (Lines 5–10). In each execution, we apply a randomly selected visual corruption pattern $\mathcal{G}$ and we corrupt the nominal output to obtain $\mathcal{G}(I \circledast f)$. We corrupt only $(I \circledast f)$ because a corruption in the nominal computation may cause an erroneous output to be produced, while corruptions affecting the approximate replica or the FIE would not be propagated to the output (refer to Fig. 2). Finally, we compute the corresponding fault impact m as in Eq. (2) by comparing $(I \circledast f)$ and $\mathcal{G}(I \circledast f)$ using the selected IQA metric $\mathcal{M}$.

Training the Regression Model

At the core of FIE we need to solve a regression problem for estimating the fault impact m (2) from the feature vector $\mathbf{x}$. Since there is no analytical expression relating m and $\mathbf{x}$, we resort to ML models [10], and train a regression model over the training set TR (Eq. (6)). In order to prepare FIE to predict multiple IQA metrics simultaneously we can either train multiple scalar regression models or, alternatively, a single multivariate regression model. In our experiments we consider both NNs and LMs, but since the proposed FIE is a general methodology, other regression models can be in principle adopted.

Setting the Detection Threshold

The Γ threshold is key to determine whether the processed images are usable or not. When a reference downstream application is available, it is possible to annotate these images as usable / unusable and compute false positives and false negatives. The former are unusable outputs not detected by the FIE, the latter are usable images (slightly or not corrupted) that are wrongly detected by the FIE. The domain expert can then select a threshold that balances the false positive rate and the false negative rate, taking into account fault management overheads.

However, since FIE produces estimates of the fault impact, it is also possible to set Γ by inspecting a set of heavily and slightly corrupted images, preparing illustrations similar to Fig. 1. Visual inspection enables a designer to manually set Γ considering the impact of faults that they would like to be reported, even without a specific payload application as a reference.

Finally, it is always possible to set the threshold Γ to control the amount of false positives, as these would trigger costly fault-management procedures. In these cases, Γ is set empirically by computing the fault impact estimates over a set of fault-free outputs and choosing the value of Γ yielding the desired percentage of false positives. Remarkably, the two latter options do not require any form of annotation or supervision over the training set TR.

5 FIE Extensions

The proposed FIE is a general methodology, which can be customized to train a classifier determining whether an output is usable or not, as long as training samples are provided from an application [2] or a domain expert. In these cases, the proposed efficient monitoring scheme pairing a nominal and approximated replica would take place, yielding the same feature vector which is associated to a *usable* or *unusable* label, rather than the measure of the fault impact (2). A binary classifier can then be trained to directly determine whether the output is usable or not.

Nonetheless, we believe that the greatest advantage of the FIE is it can be trained without supervision, which makes it very practical to train and use, since the detection scheme can be manually adjusted on the estimated severity of the fault. In what follows we describe other relevant extensions on basic scheme described in Section 3.

5.1 Adaptation

Every FIE module has been trained for a specific a specific setting[2] $(\mathcal{I}, f, a)$. Should a or f change (e.g. when a different level of approximation is required or a different filter is employed) or should $\mathcal{I}$ not be representative of the images to be processed during operations (e.g. when a different sensor or different spatial resolutions are used), then the regression model would need to be re-trained. This operation is not very practical, because it requires to repeat the design process from scratch on the new setting $(\mathcal{I}', f', a')$, in particular to generate a new training set $TR' = TR(\mathcal{I}', f', a', Q')$.

FIE allows a very practical form of adaptation to use a previously trained regression model (say on $TR(\mathcal{I}, f, a, Q)$) in different settings by solely recomputing the quantiles Q' used in the features extraction. Ideally, in fault-free conditions, images processed in the same setting $(\mathcal{I}, f, a)$ are characterized by similar distributions of D, and consequently will lead to similar feature vectors. When the type of images in $\mathcal{I}$ or the filters (f, a) change, instead of training a new model, we can adapt the feature extraction phase to the new setting $(\mathcal{I}', f', a')$ by simply computing the quantiles

2. This training also depends on the IQA metric $\mathcal{M}$, but in this section we assume the IQA metric to be fixed.

Q' from a few dozens of fault-free images. Then, a pre-trained model R trained on the initial TR can be used on the features $\mathcal{E}_{f',a',Q'}(\mathcal{I}')$.

Such a technique can be seen as a form of Transfer Learning (TL) [11], which refers to methods and algorithms for adapting a model configured for a specific task/environment to a different one. As experiments will show, this is an effective strategy to avoid acquiring large datasets to train the regression model from scratch, which might even be impossible when the application scenario deviates / evolves from the planned one.

5.2 No Approximation

In the extreme case where a is a 1×1 filter, the FIE provides fault management capabilities without the need of any replica, by estimating the IQA metric from the input image I and the (potentially corrupted) $\mathcal{G}(I \circledast f)$. This substantially reduces the overall operations to feature extraction and execution of the regressor. Results in Section 6 show that this solution is not as powerful as when the approximate replica is computed, but it is effective, and could be of interest in particularly cost-aware conditions that only require a limited degree of reliability. Moreover, this result indicates that adopting FIE is viable even when the convolution $I \circledast f$ is computed in Fourier domain, where it would be more difficult to define an approximate replica.

5.3 Extensions Beyond Convolution

Our experiments show that the FIE achieves interesting performance in 1×1 approximation where the fault impact is estimated comparing the output of a convolution block with its very same input. This fact suggests that FIE can be applied to any processing block, without having to define an approximated replica, opening the FIE to a broad range of applications, including monitoring blocks performing iterative image processing operations (e.g., RL deconvolution [12] [13]). In these latter cases, the FIE can also be used to compare the output the image produced at some intermediate stages, as this can reduce the approximation level and at the same time locate the iteration when the fault has occurred. Of course, when monitoring filters or other processing blocks that do not provide a natural image as output, it is important to make sure that the selected IQA can appropriately be used over the specific type of output.

6 Experimental Results

Our experimental campaigns have a twofold mission; first, we demonstrate that the FIE efficiently provides accurate estimates of the fault impact, and then we show that the estimated IQAs can be effectively used in fault management strategies that assess whether the output image is usable/unusable.

6.1 Experimental Setup

Computing platform

The target computing platform is composed of a single-core processor running a single-threaded image processing application. The proposed strategy is applied at the software level and the three tasks (nominal processing, approximated processing, and fault impact estimation) are executed sequentially, according to a time-triggered schedule. In the experiments we consider two different processor types for embedded and mobile applications; ARM A7 and ARM A15, having an in-order and out-of-order architecture, respectively. An Odroid XU3 board is used to run the experiments for timing performance evaluation; frequencies of the two cores were set to 1.4GHz and 2.0GHz respectively. In the future, we will also explore the same approach with different platforms, such as GPUs, to see if further improvements can be obtained.

Software Implementation of 2D-Convolution

The convolution has been implemented in C as a stand-alone application reading images from a flash drive. Three 11×11 nominal filters have been considered, namely a Gaussian smoothing filter G_{11} with $\sigma = 5$, a uniform smoothing filter U_{11} and a Laplacian of Gaussian (LoG) contrast enhancement filter L_{11} with $\sigma = 1$. We denote the approximate filters by G_h, U_h and L_h, where h is the filter size. Filter approximation is performed via filter size reduction, which enables a flexible tuning of the level of approximation, using different sizes to balance output precision and processing time. In particular, smoothing filters can be approximated by taking the inner $h \times h$ section of the nominal filter and rescaling this to sum to 1, while the approximate LoG filters are generated as $h \times h$ filters with the same σ parameter.

Design Methodology

The design methodology is implemented with two macro-modules; a state-of-the-art fault injector for microprocessor systems, LLFI [14], and a set of Matlab scripts that automate the rest of the flow in Fig. 6. We first ran a fault injection campaign on $10,000$ images from the INRIA dataset (presented in Section 6.2) following the procedure described in Section 4, then we implemented functions to inject corruption patterns, thus simulating the effects of the faults yielded by the fault injection, as in [2]. Matlab is also used to generate the training set TR and to run the experiments assessing the FIE performance.

IQA Metrics

We show the effectiveness and generality of the FIE in estimating three IQA metrics featuring different properties. The first one is RMSE, a classical error measure in statistics and signal processing. The second, SSIM index [3] measures the input similarity taking into consideration structural information, luminance and contrast. Finally, we consider GSM [8], which takes into consideration the gradient information and focuses on distortions. While SSIM index and GSM return a value between 0 and 1 (1 being the best), RMSE returns a value in $[0, 255]$ (0 being the best) which we re-scale to the interval $[0, 1]$.

Regression Models

In our experiments we test two regression models: a NN and a linear model. There is a large freedom in the design of the NN architecture and we use a feed-forward fully-connected

neural network where we set the number and the size of the hidden layers depending on the dimension d of the input features. The first hidden layer has a maximum of 64 neurons, but when d is small, the number of neurons becomes the largest power of 2 smaller or equal to d. Each subsequent hidden layer has half of the neurons of the previous layer. The last hidden layer has 2 neurons while the output layer has only 1 (the predicted fault impact). We conveniently use a saturated linear function as activation function for the last layer, yielding a regression network in the range $[0, 1]$, since all the IQA metrics have been accordingly rescaled before training. The NN is then trained over the training set TR in (6) via standard stochastic gradient descent [10]. All the results have been averaged over 5 different trained networks that differ only for their random initialization of weights.

We also use a linear model for regression, which is a simpler model. To promote interpretability of the results – which might be of interest in more critical applications – we fit the parameters of this model by promoting sparsity through the Least Absolute Shrinkage and Selection Operator (LASSO) [15]. The peculiarity of LASSO is that it automatically performs feature selection by dropping the coefficients associated to the less useful components, identified at training time. Since LASSO is deterministic, there is no need to average the results over multiple runs.

6.2 Datasets

We considered two different sources of satellite images: the first one [16] is a repository of images of US and European cities acquired at a space resolution of $0.3^{m}/_{px}$. These images were cut into 500×500 grayscale images, forming a dataset of 31,000 images, which we will refer to as the *INRIA dataset*. The second source is Microsoft Bing Maps [17], which provides an API to download aerial images for a query location and resolution. To assess the flexibility of the proposed approach, we generated three additional datasets, $BING_H, BING_M$ and $BING_L$ each containing 14,400 500×500 grayscale images each, acquired from different cities at various spatial resolution levels, $0.15^{m}/_{px}$, $0.3^{m}/_{px}$ and $1.19^{m}/_{px}$, respectively. Faulty images in both INRIA and Bing datasets have been generated using Matlab functions reproducing the visual corruption patterns that we previously identified. We split these datasets in a training set and a test set.

We also assess the FIE performance on faulty outputs generated using LLFI [14]. We randomly select 5 input images from the INRIA dataset and we repeatedly filtered them injecting single faults to generate about 5000 outputs affected by the SDCs. We compute the fault impact as in Eq. (2) using a fault-free output and use this value to compute prediction error. We refer to this third dataset as the *LLFI dataset*.

6.3 Figures of Merit

Let $\mathcal{I} = \{I_k\}$ be a set of images, let m_k be the impact of the fault on image I_k computed as in Eq. (2), and let $\widehat{m}_k$ be its estimate provided by the FIE as in Eq. (3) for a selected IQA metric $\mathcal{M}$. To assess the accuracy of FIE estimates, we compute the following Mean Absolute Error (MAE):

$$MAE = \frac{1}{\#\mathcal{I}} \sum_{k=1}^{\#\mathcal{I}} |m_k - \widehat{m}_k|. \tag{7}$$

We also assess the performance of the fault-detection scheme in Fig. 2, by setting a threshold Γ and considering an output $\mathcal{G}(I \circledast f)$ *unusable* when $m_k < \Gamma$, and *usable* otherwise. We detect as unusable each input yielding $\widehat{m}_k < \Gamma$, and we refer to a positive (negative) detection when the output is unusable (usable). The number of True Positives (TP), False Positives (FP) and False Negatives (FN) for a fixed value Γ are:

$$\begin{aligned} TP(\Gamma) &= \#\{I_k \in \mathcal{I} | m_k < \Gamma \wedge \widehat{m}_k < \Gamma\}, \\ FP(\Gamma) &= \#\{I_k \in \mathcal{I} | m_k \geq \Gamma \wedge \widehat{m}_k < \Gamma\}, \\ FN(\Gamma) &= \#\{I_k \in \mathcal{I} | m_k < \Gamma \wedge \Gamma \leq \widehat{m}_k < 1\}, \end{aligned}$$

which are used to compute the following figures of merit:

- the FIE *precision*, i.e. the percentage of unusable images over all the detections:

$$Precision(\Gamma) = \frac{TP(\Gamma)}{TP(\Gamma) + FP(\Gamma)}, \tag{8}$$

- the FIE *recall*, i.e. the percentage of detected unusable images:

$$Recall(\Gamma) = \frac{TP(\Gamma)}{TP(\Gamma) + FN(\Gamma)} \tag{9}$$

To measure the cost reduction of FIE with respect to DWC solutions, we compute the overall execution time of each hardened solution and the fault management *overhead*, i.e. the percentage of time required for fault detection w.r.t. the time of nominal processing:

$$Overhead = \frac{T_{Replica} + T_{Decision}}{T_{Nominal}}. \tag{10}$$

Each figure of merit is computed for all analyzed IQAs, considering different approximation levels for the filters, and different numbers of extracted quantiles. For the sake of space, in-depth analysis results are reported for a selected IQA metric and for the most promising combination of approximation level and number of quantiles.

6.4 IQA to Support Image Usability

We first show that an IQA metric can be effectively used to determine whether filtered images are usable or not. To this purpose, we consider an application scenario where we classify pixels in aerial images as "Natural" or "Man-made" (buildings, roads, parking lots, etc...), by means of a feed-forward NN made of 4 layers. This NN maps a filtered image $I \circledast f$ to a binary mask $B(I \circledast f)$ of the same size where 0 (resp. 1) corresponds to pixels classified as "Natural" (resp. "Man-made"). We count the number of pixels where the classification outputs $B(\mathcal{G}(I \circledast f))$ and $B(I \circledast f)$ differ and consider each faulty image $\mathcal{G}(I \circledast f)$ as "Usable" when their difference is less than 5% of the pixels and "Unusable" otherwise. We then use these labels to split a set of faulty filtered images in two subsets of usable and unusable outputs respectively. Fig. 7 shows that the SSIM is able to discriminate between usable and unusable images

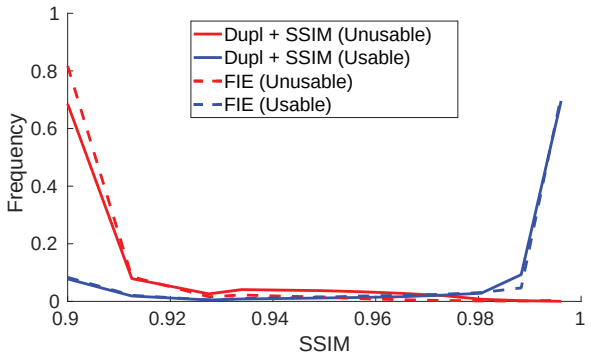


Figure 7: Distributions of the SSIM-based fault impact values for the Usable and Unusable images. For visualization purposes values lower than 0.9 are in the leftmost bin.

(blue and red lines, respectively) and that the proposed FIE (dashed lines) closely approximate the fault impact that can be computed by exact replication (solid lines).

6.5 Fault Impact Estimation Analysis

We further assess the accuracy of the FIE by measuring the MAE on the metric estimation (7) and show that the FIE can provide accurate estimates of the IQA metrics. Thus, FIE can replace the computation of fault impact as in Eq. (2) via exact replication. Fig. 8 shows the results of the SSIM index estimation and the fault management overhead in the $(INRIA, G_{11}, G_h)$ setting, for all the combinations of approximation level h and number of quantiles d. As expected, when decreasing the approximated filter size, the MAE increases (solid lines), while the overhead decreases (dashed lines). We select $d = 38$ quantiles as a good trade-off in terms of accuracy and computational costs, and use this configuration for the analysis in Tab. 1 and Tab. 2 where we test different approximation levels.

Tab. 1 presents the MAE values of the estimation of the considered IQA metrics when using both an NN and a linear model fit using LASSO. Results show that FIE can reliably estimate the metric values with nearly negligible errors on the set of fault-free images. This is very important to control false positives in fault detection schemes. On corrupted images, the FIE behaves as expected, only marginally decreasing its accuracy. MAE values indicate that NNs always outperform linear models, which are indeed simpler models. The feature selection capability of LASSO indicates that the most important quantiles for estimating the fault impact (corresponding to nonzero coefficients in the solution) are those associated to the center and the tails of the distribution. Fixed quantiles $\{w_i\}$ are not primarily relevant, but we experienced these are very important to handle a few corruption patterns. Due to space limitation, we do not provide further analysis on the intepretability, and in the next sections we consider only NNs as regression models. It is worth mentioning that we trained and tested FIE also on Laplacian filters presented in Section 6.1 achieving similar results as the Gaussian and the Uniform filters (see Section 6.7). This shows that the proposed methodology successfully applies also to non-smoothing filters.

Tab. 2 reports both the execution times and computation overhead on the target platforms of the FIE at various approximation levels as well as those of the baseline alternatives presented in Section 2.4, i.e., the *flexible DWC* and the *Dupl+IQA*, which computes the IQA metric after replication. For the sake of comparison, we also report the execution times of the nominal processing. For a more detailed analysis, we report in Tab. 3 the execution times of the all the software modules considered in the various versions of the filter and checkers. These values confirm that approximating the filter replica offers a considerable time saving. Moreover, the execution times of GSM and SSIM index are not affordable, while the execution time of the FIE checker, even if longer than the ones of the DWC and RMSE checker, is well compensated by the savings of adopting an approximated replica. All in all, FIE attains the lowest computational overhead among the considered solutions; in particular, it improves the execution time up 44% w.r.t. the DWC baseline, still preserving accurate estimates. It is worth noting that – compared to alternatives based on DWC – the FIE can guarantee additional computational savings by preventing the re-execution of the whole pipeline on images deemed as usable.

As an additional remark, we comment that FIE solutions are also rather efficient to train and configure. The preparation of a training set containing 14,400 images took about 1 hour on a standard workstation. NNs training time depends on the network architecture and the number of quantiles: in our setup, it spans from a few seconds to about 1 hour. LASSO always took a few seconds. It is worth mentioning that features do not need to be re-extracted when changing the IQA metric or the number of adopted quantiles, hence reducing the total time required to configure the FIE with a different configuration. Exploring all FIE configurations with respect to IQA metrics, approximation levels, number of quantiles and regression models took less than 30 hours.

Finally, we emphasise the remarkable performance of the FIE based on the 1x1 approximation, where we directly compare input and output. This solution has minimal computational overhead (as shown in Tab. 2), but still provides accurate fault impact estimates.

6.6 Fault Detection Analysis

In this second experimental campaign we have exploited the output of the FIE module to detect heavily corrupted images (i.e. images such that $m < \Gamma$) to be discarded and reprocessed. As discussed in Section 4, there are different ways of setting the detection threshold Γ, either depending on the minimum visual impact of faults to be detected, or to yield an the expected false positive rate. In Tab. 4 we pursue this latter option and demonstrate that, for all the considered values of Γ, the detector achieves high precision (ranging from $94.58\% - 96.70\%$) and recall (ranging in $(88.2\% - 97.8\%)$ depending on the approximation levels. This indicates that FIE turns to be very accurate when classifying images in usable/unusable. These results concern the FIE based on the SSIM in two settings with $d = 38$: $(INRIA, G_{11}, G_5)$, where a 5×5 approximated filter is used, and $(INRIA, G_{11}, G_1)$, where no approximation is employed. Both settings yield a good trade-off between the estimation error and the computational cost.

Table 1: MAE of the estimation of various IQA metrics in the setting $(INRIA, G_{11}, G_h)$ for $h \in \{1, 3, 5, 7, 9\}$ and $d = 38$.

Approx	Using NN as regressor						Using LASSO as regressor					
	MAE (fault-free images)			MAE (corrupted images)			MAE (fault-free images)			MAE (corrupted images)		
	SSIM (x10^{-2})	RMSE (x10^{-2})	GSM (x10^{-2})	SSIM (x10^{-2})	RMSE (x10^{-2})	GSM (x10^{-2})	SSIM (x10^{-2})	RMSE (x10^{-2})	GSM (x10^{-2})	SSIM (x10^{-2})	RMSE (x10^{-2})	GSM (x10^{-2})
1x1	1.236	1.170	0.091	1.923	0.991	0.198	3.412	1.727	0.247	3.886	2.376	0.356
3x3	0.608	0.599	0.050	1.367	0.655	0.163	2.549	1.695	0.175	2.686	2.007	0.271
5x5	0.347	0.465	0.045	1.035	0.563	0.147	1.433	1.449	0.093	1.773	1.821	0.211
7x7	0.274	0.503	0.020	0.898	0.653	0.133	0.681	1.291	0.051	1.222	1.755	0.183
9x9	0.198	0.444	0.012	0.804	0.713	0.125	0.166	1.180	0.029	1.406	1.973	0.198

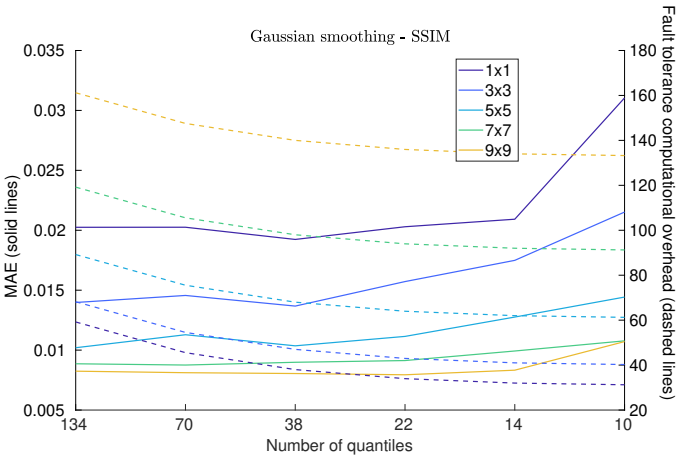


Figure 8: MAE for the estimation of the SSIM index (solid line) and computational overhead (dashed line) for different number of quantiles and different approximation levels.

Table 2: Computational overhead of the FIE and of the baseline alternatives.

Scenario	Approx	*A*7 core		*A*15 core	
		Total (ms)	Overhead	Total (ms)	Overhead
Nominal filter	-	158	-	58	-
Dupl+GSM	-	1326	739%	523	801%
Dupl+SSIM	-	1268	703%	410	607%
Dupl+RMSE	-	320	103%	117	102%
Flexible DWC	-	319	102%	117	102%
FIE	9x9	301	91%	110	90%
FIE	7x7	259	64%	100	72%
FIE	5x5	229	45%	89	53%
FIE	3x3	208	32%	78	34%
FIE	1x1	178	13%	68	17%

Table 3: Execution times of the various software modules.

Module	Execution time (ms)	
	*A*7 core	*A*15 core
Filter 1x1	0	0
Filter 3x3	30	10
Filter 5x5	51	21
Filter 7x7	81	32
Filter 9x9	123	42
Filter 11x11	158	58
FIE module	20	10
GSM	1010	407
SSIM	952	294
RMSE	4	1
Flexible DWC checker	3	1

On top of the very high classification performance, we report the MAE as in Eq. (7) but averaged over all the erroneous detections, namely FP and FN, and notice these corresponds to very small errors. This is a very important aspect, since these errors can be taken into account when setting the threshold Γ based on the maximum fault impact that can be accepted.

Tab. 4 indicates that FIE achieves similar precision and recall values on the LLFI dataset even though the MAE is larger than in the INRIA dataset. This small performance drop is due to the fact that FIE was trained over different visual corruption patterns, and confirms that the proposed solution can be reliably employed also in unseen scenarios, achieving good detection performance.

6.7 FIE Adaptation Analysis

Our last experimental campaign investigates the effectiveness of the adaptation strategy described in Section 5.1 when filters f, a change and when the spatial resolution of images changes. To this purpose we consider three different FIE solutions: *i*) the *ideal* one, where the FIE is entirely trained to operate in the new setting characterized by the new filters f', a' or the new images $\mathcal{I}'$. *ii*) The *adaptive* FIE, which is configured on an initial training set $TR(\mathcal{I}, f, a, Q)$ and, to better operate in the target setting, adapts the set of quantiles $Q \rightarrow Q'$ and the feature extraction phase $\mathcal{E}_{f',a',Q'}$ accordingly. *iii*) The *no adaptation* solution, where FIE was trained on the training set $TR(\mathcal{I}, f, a, Q)$ and is used *as is* in the target settings. We restrict our analysis to 5×5 and 1×1 approximation.

Tab. 5 shows the estimation accuracy of the three solutions when changing the filters $(f, a) \rightarrow (f', a')$ from Gaussian $G_{(\cdot)}$ to Uniform $U_{(\cdot)}$ and vice-versa. We notice that, most often, the estimation error without adaptation substantially increases, while the proposed adaptation achieves as good performance as in the *ideal* solution that was trained using the new filters. When changing from $G \rightarrow U$, adaptation provides substantial improvements on faulty images and smaller improvements on fault-free images, where we remark it is very important to provide accurate estimates not to have large number of false alarms. When changing $U \rightarrow G$ we observe that the no-adaptation solution is surprisingly better than the ideal one in fault-free images, and we explain this phenomenon as a consequence of a tightening of the distribution of pixel-wise differences in D. When this happens and the quantiles are not adjusted to the new distribution, the components of the feature vector are less spread, describing a less-corrupted output. This results

Table 4: Performance of the FIE for the SSIM estimation, having set Γ to yield a fixed FPR. The setting are $(INRIA, G_{11}, G_5)$ (above) and $(INRIA, G_{11}, G_1)$ (below) with $d = 38$.

Approximate filter size: 5x5									
		INRIA dataset				LLFI dataset			
% FP (fault-free)	Γ	Precision	Recall	MAE (x10^{-2}) Average	Std	Precision	Recall	MAE (x10^{-2}) Average	Std
1.00 %	0.957	96.48 %	95.65 %	4.676	4.956	94.05 %	95.59 %	24.420	25.671
0.75 %	0.953	96.61 %	95.53 %	4.762	5.075	99.77 %	95.40 %	30.376	30.019
0.50 %	0.946	96.70 %	95.37 %	4.901	5.322	100.00 %	95.11 %	30.822	29.896
0.25 %	0.906	95.94 %	94.37 %	5.011	5.385	100.00 %	96.24 %	39.345	30.876
0.10 %	0.772	96.38 %	97.85 %	6.881	7.314	99.83 %	93.58 %	32.417	27.598

Approximate filter size: 1x1									
		INRIA dataset				LLFI dataset			
% FP (fault-free)	Γ	Precision	Recall	MAE (x10^{-2}) Average	Std	Precision	Recall	MAE (x10^{-2}) Average	Std
1.00 %	0.871	94.58 %	88.26 %	10.778	9.959	99.92 %	85.70 %	33.364	17.540
0.75 %	0.837	95.43 %	91.49 %	13.687	10.875	100.00 %	85.69 %	34.428	17.247
0.50 %	0.733	94.81 %	94.34 %	13.948	12.887	99.61 %	85.42 %	35.117	17.640
0.25 %	0.639	95.49 %	95.24 %	12.734	14.218	97.89 %	88.66 %	34.101	21.978
0.10 %	0.528	96.58 %	96.46 %	10.838	14.637	98.51 %	87.44 %	31.138	24.069

in a lower prediction error, since the IQA value associated to any faulty-free image is equal to $B_{\mathcal{M}}$. The advantages of adaptation on faulty images are very apparent in Fig. 9, where we plot the SSIM estimation MAEs of the Gaussian to Uniform experiment for both the $h = 1$ and $h = 5$ settings and for all the number of quantiles d considered in our experiments.

Tab. 6 presents the estimation error in the scenario where the spatial resolution of test images $\mathcal{I}'$ is different than images $\mathcal{I}$ used for training. In particular, we train our FIE on $BING_M$ and explore both the cases where the spatial resolution decreases ($BING_M \rightarrow BING_L$) and increases ($BING_M \rightarrow BING_H$). A change in the population of test images, like that in spatial resolution, is perhaps more interesting than the filter change as the latter could be handled at design time. In contrast, changes in the image population might not be foreseen at design time, or it would not be possible to gather enough images for training in the new settings. Thus, adaptation might be the only viable option. Tab. 6 shows that when the resolution decreases, the proposed adaptation grants a substantial performance improvement. On the other hand, when moving to higher spatial resolution, we observe the same tightening of the distribution of pixel-wise differences in D, as in the previous experiment, leading to the same behavior of the no-adapt solution. It is worth mentioning that also in this case adaptation preserves the estimation accuracy when there the change would result in a substantial performance drop.

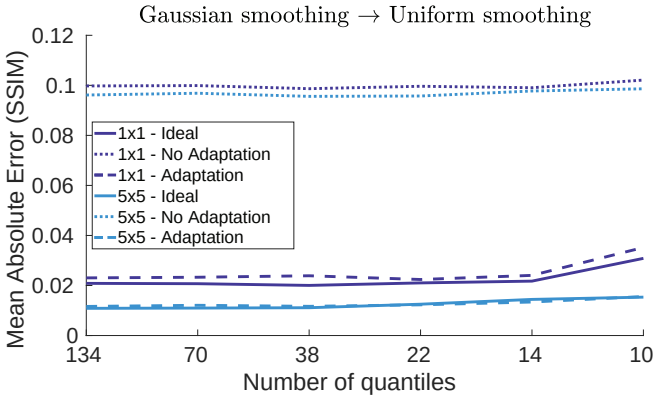


Figure 9: MAE of the SSIM index estimation in the filter-change TL scenarios from $(INRIA, G_{11}, G_h)$ to $(INRIA, U_{11}, U_h)$ for $h \in \{1, 5\}$.

7 Related Work

Approximate Computing has been widely employed in the past years to define lighter redundancy-based hardening schemes at various abstraction levels: at logic level, Register-Transfer Level (RTL) and system level, and we here review the most relevant ones, from the ones working at the lowest abstraction level to those at the highest one.

Various approaches (e.g. [18], [19], [20], [21]) defined approximate Triple Modular Redundancy (TMR) schemes at logic level. The basic scheme is defined for logic circuits and applied to each single-bit output combinatorial function with the aim of minimizing the trade-off between the overall area and the error rate introduced by approximation. Various automated synthesis approaches [18], [19] have been proposed adopting this solution, whereas the scheme in [21] also approximates the nominal replica by using a Boolean factorization method. Finally [20], [22] target FPGA devices, thus specifying also an approximated TMR that is aware of the technological mapping.

In a similar direction, approximate Concurrent Error Detection (CED) schemes [23] and voters [24] for combinatorial logic circuits have been investigated. Approximate CED schemes are further enhanced in [25] to obtain two concurrent error masking approaches for logic and timing errors. Finally, approximated checker and voter architectures for inexact DWC and TMR have been proposed in [24], to optimize power consumption and circuit area and tolerate both process and supply voltage variations.

Moving to RTL, it is possible to classify existing solutions into two groups: approaches that exploit numeric precision reduction and those that use an approximated replica of the nominal block. The basic idea of the Reduced Precision Redundancy (RPR) is to define replicated components that

Table 5: FIE adaptation to filter-change ($d = 38$) for different IQA metrics. Networks NN trained on $TR(INRIA, G_{11}, G_h, Q)$ are used for fault estimation in the settings $(INRIA, U_{11}, U_h)$ (above) and viceversa (below).

IQA metric	Approx	$(INRIA, G_{11}, G_h) \rightarrow (INRIA, U_{11}, U_h)$						$(INRIA, U_{11}, U_h) \rightarrow (INRIA, G_{11}, G_h)$					
		MAE (faulty)			MAE (fault-free)			MAE (faulty)			MAE (fault-free)		
		ideal	no adapt	adapt	ideal	no adapt	adapt	ideal	no adapt	adapt	ideal	no adapt	adapt
SSIM	1x1	2.003	9.866	2.389	1.378	1.606	2.027	1.923	9.670	1.993	1.236	1.099	1.156
($\times 10^{-2}$)	5x5	1.111	9.559	1.167	0.448	1.123	0.302	1.035	9.164	1.219	0.347	0.158	0.719
RMSE	1x1	1.053	3.949	1.263	1.231	1.338	1.299	0.991	3.890	1.194	1.170	1.060	1.115
($\times 10^{-2}$)	5x5	0.618	3.689	0.714	0.551	0.859	0.459	0.563	3.558	0.806	0.465	0.359	0.713
GSM	1x1	0.211	0.705	0.226	0.114	0.118	0.139	0.198	0.701	0.227	0.091	0.093	0.094
($\times 10^{-2}$)	5x5	0.141	0.698	0.155	0.038	0.097	0.039	0.147	0.660	0.158	0.045	0.015	0.050

Table 6: FIE adaptation to dataset-change ($d = 38$) for different IQA metrics. Networks NN trained on $TR(BING_M, G_{11}, G_h, Q)$ are used for fault estimation in the settings $(BING_L, G_{11}, G_h)$ (above) and $(BING_H, G_{11}, G_h)$ (below).

IQA metric	Approx	$(BING_M, G_{11}, G_h) \rightarrow (BING_L, G_{11}, G_h)$						$(BING_M, G_{11}, G_h) \rightarrow (BING_H, G_{11}, G_h)$					
		MAE (faulty)			MAE (fault-free)			MAE (faulty)			MAE (fault-free)		
		ideal	no adapt	adapt	ideal	no adapt	adapt	ideal	no adapt	adapt	ideal	no adapt	adapt
SSIM	1x1	1.389	9.250	3.180	0.443	13.626	0.555	1.140	1.195	1.975	0.594	0.306	0.877
($\times 10^{-2}$)	5x5	0.711	2.469	1.722	0.121	2.509	0.156	0.813	0.824	1.094	0.215	0.175	0.243
RMSE	1x1	0.591	3.452	1.463	0.529	5.362	0.856	0.507	0.556	1.171	0.517	0.464	0.772
($\times 10^{-2}$)	5x5	0.318	0.932	0.540	0.272	1.297	0.444	0.531	0.527	0.695	0.369	0.344	0.440
GSM	1x1	0.130	0.853	0.305	0.032	1.061	0.055	0.129	0.141	0.199	0.043	0.029	0.082
($\times 10^{-2}$)	5x5	0.098	0.298	0.194	0.015	0.254	0.024	0.112	0.119	0.146	0.020	0.015	0.024

elaborate only on a subset of the most significant bits processed by the nominal system. The outcome is that it is possible to confine errors in a subset of the least significant bits of the scalar values in output, to be tuned according to the desired precision. This idea is exploited in several approaches enhancing TMR [26], targeting FPGA devices [27] or focusing on signal processing applications [28].

The works proposed by Shanbhag *et alii* [28], [29], [30] fall into the second group, and deal with signal processing applications. The proposed underlying strategy is called Algorithmic Noise Tolerance (ANT) (refined in the subsequent publications), and associates the nominal computation with an estimator of the same functionality. During the operational phase of the system, if the difference between the processed signal and the estimator one is above a set threshold (the acceptable noise) a problem is detected. The goal of the approach is to detect timing errors due to the reduction of the threshold voltage (applied for power saving). In [29] the strategy does not use replicas, but compares the input and output of the filter taken into account, whereas in [28], RPR is also introduced to extend the initial approach. Later on, in [30], the same authors demonstrated the suitability of the approach also for soft errors tolerance in the combinatorial logic and they extended the basic ANT scheme to mimic N-Modular Redundancy (NMR) to achieve fault tolerance.

All these approaches are based on the analysis of the single processed data, being it the input of a logic circuit, or single value of the input signal at a given moment. More precisely, these methodologies are based on the idea of determining whether a component of the information being processed (a bit, a pixel or, more in general, a value) is affected by an error by analysing the component itself. In our approach, we move from the analysis of the single value to the entire data, that is from the single pixel to the entire image. As a result, even though the image has erroneous pixels, the acceptability property applies to the whole data rather than its components, based on the impact of the fault. On the other hand, an interesting aspect of ANT and [31] is the exploitation of the intrinsic error tolerance of the target application to define the hardening scheme, and in particular the estimated, reduced replica and the checker. In this perspective, ANT is, to some extent, the work that is more similar to our proposal, although not directly applicable to our context.

At system level there is a single work exploiting approximation for fault tolerance, namely [32]. The paper proposes a design approach that combines an application-level hardening techniques based on NMR and the subsequent mapping and scheduling of the applications' tasks on the available processing units. The peculiar aspect is the use of approximation on replicated tasks to reduce their execution time. Since the output of the replicated tasks are "similar" but different (because of approximation), a classical majority voting cannot be applied, however the authors do not discuss how it can be designed.

A different class of approaches exploiting some usability concepts in image processing, includes solutions that optimize other figures of merit, such as energy saving (e.g., [31]), low-power (e.g., [33]), and cope with the reduced quality of the processed image, by proposing *error tolerant* solutions to the inexactness introduced by the adopted strategies (e.g., aggressive voltage scaling in [31]). Although the same idea of usability is somehow exploited, these approaches offer and provide hints that cannot be used or compared against in our scenario.

Finally, another work abandoning the correct/incorrect paradigm to move towards the usable/unusable one is the work presented in [2], where the classical DWC scheme is modified to use a Convolutional Neural Network (CNN) to determine whether an error corrupted the image (or an intermediate result in case of an application pipeline)

instead of the rigid TRC. Because no approximation is used, two identical replicas of the processed image are compared to determine whether they can be considered usable, and a supervised learning approach is adopted to train the CNN. The CNN overcomes the limitations of the SSIM index from the performance point of view, however it requires the definition of a tailored concept of usable/unusable and of supervised learning, limitations that the proposed solution avoids, offering a more flexible and considerably easier design and training process.

8 Conclusions

We present the FIE, a new framework for performing fault detection in a flexible and lightweight manner. The peculiarity of the FIE is to quantitatively assess the visual impact of the fault, by estimating an IQA metric that can reliably distinguish faults that severely affect the output from small distortions. This makes it the ideal module in detection schemes that refer to the usability of the output rather than its exactness, with an additional saving of computational time and power during fault management. The FIE module is trained in a completely unsupervised manner, and is very flexible and easy to adapt to different settings. Moreover, the FIE can be straightforwardly modified to directly assess usability, as long as annotations are provided from an application or domain expert. Our experiments demonstrate that the FIE achieves a reduction in terms of execution times up to 44% with respect to the classical DWC, yielding very high precision and recall.

As future work, we will develop further our approach to monitor more general applications, including iterative filters and deep learning models. The major challenge is the definition of sound fault assessment metrics in situations where IQA metrics cannot be adopted. Another promising direction to investigate is the design of detection schemes combining multiple IQA metric estimates, as these can be computed from the same feature vector. Furthermore, considering the numerous aspects driving the design and tuning of the FIE, characterized by different levels of accuracy and performance improvements, we envision a design space exploration framework, to identify through multi-objective optimization the different alternative solutions, among which the expert can choose the most appropriate one, according to his/her constraints and goals.

Acknowledgment

This work has been partially supported by Intel Corporation under the grant award titled "Adaptive Application-oriented Fault Detection for Reliable Image Processing". G. Boracchi gratefully acknowledges the support of NVIDIA Corporation with the donation of the Titan Xp and Titan V GPUs that we have used in this research. D. Stucchi's PhD Grant is partially supported by Epta company.

References

[1] R. L. Davidson and C. P. Bridges, "Error resilient gpu accelerated image processing for space applications," *IEEE Trans. Parallel and Distributed Systems*, pp. 1–1, 2018.

[2] M. Biasielli, C. Bolchini, L. Cassano, E. Koyuncu, and A. Miele, "A Neural Network Based Fault Management Scheme for Reliable Image Processing," *IEEE Trans. Computers*, pp. 1–13, 2020.

[3] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli, "Image quality assessment: from error visibility to structural similarity," *IEEE Trans. Image Processing*, vol. 13, no. 4, pp. 600–612, 2004.

[4] A. Foi, V. Katkovnik, and K. Egiazarian, "Pointwise Shape-Adaptive DCT for High-Quality Denoising and Deblocking of Grayscale and Color Images," *IEEE Trans. on Image Processing*, vol. 16, no. 5, pp. 1395–1411, 2007.

[5] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet Classification with Deep Convolutional Neural Networks," in *Proc. Advances in Neural Information Processing Systems*, 2012, pp. 1097–1105.

[6] K.-H. Thung and P. Raveendran, "A survey of image quality measures," in *Proc. Int. Conf. Technical Postgraduates*, 2009, pp. 1–4.

[7] A. G. George and A. Prabavathy, "A survey on different approaches used in image quality assessment," *Int. Journal Computer Science and Network Security*, vol. 3, no. 2, 2014.

[8] A. Liu, W. Lin, and M. Narwaria, "Image quality assessment based on gradient similarity," *IEEE Trans. Image Processing*, vol. 21, no. 4, pp. 1500–1512, 2011.

[9] C. Bolchini, L. Cassano, A. Mazzeo, and A. Miele, "Error Modeling for Image Processing Filters accelerated onto SRAM-based FPGAs," in *Proc. Intl. Symp. on On-Line Testing and Robust System Design*, 2020, pp. 1–6.

[10] C. M. Bishop, *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer-Verlag New York, Inc., 2006.

[11] S. J. Pan and Q. Yang, "A Survey on Transfer Learning," *IEEE Trans. Knowledge and Data Engineering*, vol. 22, no. 10, pp. 1345–1359, 2010.

[12] W. H. Richardson, "Bayesian-based iterative method of image restoration," *JoSA*, vol. 62, no. 1, pp. 55–59, 1972.

[13] L. B. Lucy, "An iterative technique for the rectification of observed distributions," *The astronomical journal*, vol. 79, p. 745, 1974.

[14] J. Wei, A. Thomas, G. Li, and K. Pattabiraman, "Quantifying the Accuracy of High-Level Fault Injection Techniques for Hardware Faults," in *Proc. Int. Conf. Dependable Systems and Networks*, 2014, pp. 375–382.

[15] R. Tibshirani, "Regression shrinkage and selection via the lasso," *Journal of the Royal Statistical Society: Series B (Methodological)*, vol. 58, no. 1, pp. 267–288, 1996.

[16] E. Maggiori, Y. Tarabalka, G. Charpiat, and P. Alliez, "Can semantic labeling methods generalize to any city? the inria aerial image labeling benchmark," in *Proc. Int. Geoscience and Remote Sensing Symp.*, 2017, pp. 3226–3229.

[17] Microsoft Corporation, "Bing Maps APIs," https://msdn.microsoft.com/en-us/library/dd877180.aspx, 2018.

[18] I. Albandes, A. Serrano-Cases, A. J. Sańchez-Clemente, M. Martins, A. Martínez-Álvarez, S. Cuenca-Asensi, and F. L. Kastensmidt, "Improving approximate-TMR using multi-objective optimization genetic algorithm," in *Proc. Latin-American Test Symposium*, 2018, pp. 1–6.

[19] A. J. Sańchez-Clemente, L. Entrena, R. Hrbacek, and L. Sekanina, "Error Mitigation Using Approximate Logic Circuits: A Comparison of Probabilistic and Evolutionary Approaches," *IEEE Trans. Reliability*, vol. 65, no. 4, pp. 1871–1883, 2016.

[20] A. J. Sańchez-Clemente, L. Entrena, and M. García-Valderas, "Partial TMR in FPGAs Using Approximate Logic Circuits," *IEEE Trans. Nuclear Science*, vol. 63, no. 4, pp. 2233–2240, 2016.

[21] I. A. Gomes, M. G. Martins, A. I. Reis, and F. L. Kastensmidt, "Exploring the use of approximate TMR to mask transient faults in logic with low area overhead," *Microelectronics Reliability*, vol. 55, no. 9, pp. 2072–2076, 2015.

[22] S. Venkataraman, R. Santos, and A. Kumar, "A flexible inexact TMR technique for SRAM-based FPGAs," in *Proc. Design, Automation Test in Europe Conf.*, 2016, pp. 810–813.

[23] M. R. Choudhury and K. Mohanram, "Approximate logic circuits for low overhead, non-intrusive concurrent error detection," in *Proc. Design, Automation Test in Europe Conf.*, 2008, pp. 903–908.

[24] K. Chen, J. Han, and F. Lombardi, "Two Approximate Voting Schemes for Reliable Computing," *IEEE Trans. Computers*, vol. 66, no. 7, pp. 1227–1239, July 2017.

[25] M. R. Choudhury and K. Mohanram, "Low Cost Concurrent Error Masking Using Approximate Logic Circuits," *IEEE Trans. Computer-Aided Design of Integrated Circuits and Systems*, vol. 32, no. 8, pp. 1163–1176, 2013.

[26] A. Ullah, P. Reviriego, S. Pontarelli, and J. A. Maestro, "Majority voting-based reduced precision redundancy adders," *IEEE Trans. Device and Materials Reliability*, vol. 18, no. 1, pp. 122–124, 2018.
[27] B. Pratt, M. Fuller, and M. Wirthlin, "Reduced-Precision Redundancy on FPGAs," *Int. Journal Reconfigurable Computing*, pp. 1–12, 2011.
[28] B. Shim, S. R. Sridhara, and N. R. Shanbhag, "Reliable low-power digital signal processing via reduced precision redundancy," *IEEE Trans. Very Large Scale Integration Systems*, vol. 12, no. 5, pp. 497–510, 2004.
[29] R. Hegde and N. R. Shanbhag, "Soft digital signal processing," *IEEE Trans. Very Large Scale Integration Systems*, vol. 9, no. 6, pp. 813–823, 2001.
[30] B. Shim and N. R. Shanbhag, "Energy-efficient soft error-tolerant digital signal processing," *IEEE Trans. Very Large Scale Integration Systems*, vol. 14, no. 4, pp. 336–348, 2006.
[31] S. H. Kim, S. Mukhopadhyay, and M. Wolf, "Modeling and analysis of image dependence and its implications for energy savings in error tolerant image processing," *IEEE Trans. Computer-Aided Design of Integrated Circuits and Systems*, vol. 30, no. 8, pp. 1163–1172, 2011.
[32] F. Baharvand and S. G. Miremadi, "LEXACT: Low energy n-modular redundancy using approximate computing for real-time multicore processors," *IEEE Trans. Emerging Topics in Computing*, pp. 1–11, 2017.
[33] M. Gao, Q. Wang, M. T. Arafin, Y. Lyu, and G. Qu, "Approximate computing for low power and security in the internet of things," *Computer*, vol. 50, no. 6, pp. 27–34, 2017.