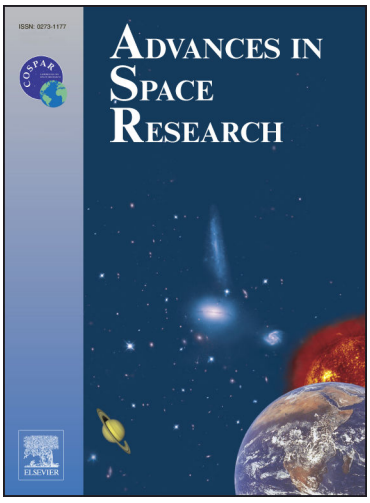
Journal Pre-proofs



Edge Computing in Space: Design of an FPGA Architecture for Thermal Anomaly Detection based on a Machine Learning Approach

Carmen MISA MOREIRA, Carl SHNEIDER, Andreas M. HEIN

PII: S0273-1177(25)00002-X
DOI: https://doi.org/10.1016/j.asr.2025.01.003
Reference: JASR 18071

To appear in: *Advances in Space Research*

Received Date: 22 April 2024
Revised Date: 17 December 2024
Accepted Date: 3 January 2025

Please cite this article as: MOREIRA, C.M., SHNEIDER, C., HEIN, A.M., Edge Computing in Space: Design of an FPGA Architecture for Thermal Anomaly Detection based on a Machine Learning Approach, *Advances in Space Research* (2025), doi: https://doi.org/10.1016/j.asr.2025.01.003

Available online at www.sciencedirect.com

ScienceDirect

Advances in Space Research xx (2025) xxx-xxx

ADVANCES IN SPACE RESEARCH (a COSPAR publication)

www.elsevier.com/locate/asr

Edge Computing in Space: Design of an FPGA Architecture for Thermal Anomaly Detection based on a Machine Learning Approach

Carmen MISA MOREIRA[a,*], Carl SHNEIDER[a], Andreas M. HEIN[a]

[a]*Interdisciplinary Centre for Security, Reliability and Trust (SnT), University of Luxembourg, 29 Av. J. F. Kennedy, Luxembourg, L-1855, Luxembourg*

Received 17 December 2024; Received in final form 17 December 2024; Accepted 17 December 2024;
Available online 17 December 2024

Abstract

The extensive range of sensors, devices, and instrumentation on onboard space systems generates a substantial volume of data intended for transmission to the ground. However, the downlink data rate is inherently constrained by transmitting power and ground station access. Edge computing aims to reduce the latency and bandwidth within a downlink by processing the data as close as possible to where it has been generated, by placing the processing hardware close to the data source. In this paper, we apply edge computing to a payload for thermal anomaly detection, developed at the University of Luxembourg. The payload encompasses a series of Forward-Looking Infrared (FLIR) high-resolution Long-Wavelength Infrared (LWIR) micro-thermal cameras as an edge-sensing component to generate the thermal images. A Field-Programmable Gate Array (FPGA) acts as an edge-computing system for processing thermal images and heat distribution profiles, using a Support Vector Machine (SVM) algorithm to detect anomalies.

Keywords: Space Systems; Edge Computing; Machine Learning; Anomaly Detection; Support Vector Machine; FPGA

1. Introduction

The relevance of data as the primary output of space missions has been underscored in the literature (Bouwmeester & Guo, 2010). However, the efficacy of communication links between spacecraft and ground stations encounters limitations influenced by diverse factors (Denby & Lucia, 2020):

1. Physical Constraints: The form factor, mass, and volume limitations of spacecraft impact power availability for communications and antenna size (Santoni et al., 2014; Babuscia et al., 2013; Buttazzoni et al., 2017).
2. Orbit and Attitude Characteristics: The orbit and attitude of a spacecraft determine the timing and frequency of data collection.
3. Earth Rotation and Ground Station Location: The interplay between orbit, Earth rotation, and ground station locations imposes constraints on link availability, duration, and bit rate.

The miniaturization trend in space systems introduces further constraints on power generation and antenna dimensions (Colin et al., 2018; Bouwmeester & Guo, 2010; Selva & Krejci, 2012). Edge computing provides on-orbit data processing capabilities, enabling the transmission of pertinent data to Earth, thereby mitigating bandwidth and data rate requirements (Zhang et al., 2022; Li et al., 2021). This paper explores the integration of edge computing with machine learning in space applications. While this combination has been implemented in terrestrial applications like mobile networks (Shakarami et al., 2020), recent research demonstrates its viability in space missions. The Φ-Sat-1 mission exemplifies the implementation of deep neural networks onboard satellites for earth observation (Giuffrida et al., 2022). Additionally, the use of machine learning techniques to facilitate global flood mapping on-

*Corresponding author: Tel.: +34-662-305-074
Email addresses: carmen.misa@uni.lu (Carmen MISA MOREIRA), carl.shneider@uni.lu (Carl SHNEIDER), andreas.hein@uni.lu (Andreas M. HEIN)

https://dx.doi.org/10.1016/j.jasr.xxxx.xx.xxx
0273-1177/

board satellites showcases the feasibility of onboard data processing for real-time applications (Mateo-Garcia et al., 2021). The HYPSO-1 project focuses on near real-time hyperspectral image classification for in-orbit decision-making, demonstrating the utilization of machine learning algorithms directly onboard satellites (Bratvold, 2022). Furthermore, initiatives like the OPS-SAT mission underscore the practical implementation of machine learning for spacecraft autonomy, highlighting advancements in onboard AI capabilities for autonomous decision-making in space (Labrèche et al., 2022). These developments also encompass real-time image processing tasks and satellite image classification using machine learning algorithms (Meoni et al., 2024; Kacker et al., 2022), emphasizing the practical challenges and innovations in deploying AI in space. The junction of edge computing and machine learning holds promise in identifying off-nominal behavior in miniaturized spacecraft, particularly when downlink limitations hinder the transmission of pertinent telemetry data (Chen et al., 2021; Nalepa et al., 2022; Ibrahim et al., 2018).

To illustrate the potential of anomaly detection in spacecraft, this paper focuses on thermal anomaly detection. Given the increasing onboard computing capacities and heat release of miniaturized spacecraft, thermal anomalies are expected to become more significant (Selva & Krejci, 2012). Temperature, a reliable indicator of equipment malfunction, is crucial in predictive maintenance, considering the correlation between elevated temperature and decreased electronic component lifespan (Laib dit Leksir et al., 2018; Lakshminarayanan & Sriraam, 2014; Jayaraman et al., 1997). Early detection of faults through thermal inspection and analysis of thermal distribution profiles enables preventive actions, averting irreversible damage to the system (Yang et al., 2019).

This paper introduces an edge computing payload adopting machine learning for thermal anomaly detection. A Field-Programmable Gate Array (FPGA) serves as the edge-computing system for onboard data processing, and a Forward-Looking Infrared (FLIR) High-Resolution Long Wavelength Infrared (LWIR) micro-thermal camera acts as the edge-sensing component for capturing thermal images. The primary objective is to establish a reliable and accurate monitoring system for fault detection in space systems attributable to thermal phenomena, enhancing the understanding of their impact on normal system behavior.

The subsequent sections of this paper are organized as follows. Section 1 provides an overview of motivation, objectives, and the drawbacks of the current bent-pipe architecture, proposing edge computing as an alternative. Section 2 delves into the components and high-level architecture of the payload. Section 3 introduces machine learning to accelerate space applications, offering a comprehensive explanation of the Support Vector Machine (SVM) algorithm and its digital implementation in an FPGA. Section 4 details the results, encompassing data processing, feature extraction, machine learning training, modeling, and run-time inference along with a comprehensive evaluation of the SVM model, followed by a dataset corruption analysis simulating faults due to the space environment. Section 5 engages in a discussion concerning the results, implications, and findings. Finally, section 6 concludes and outlines future directions.

2. Edge Computing Payload

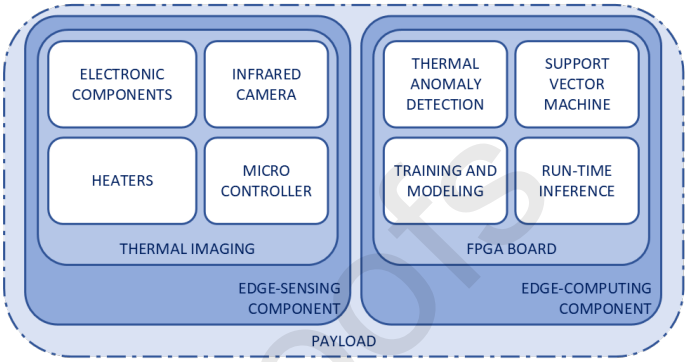


Fig. 1: High-level architecture of the payload divided into edge-sensing and edge-computing components.

The primary constituents of the payload, as illustrated in Figure 1, encompass the edge-computing and the edge-sensing components. Our methodology advocates for a machine learning approach, employing the Support Vector Machine (SVM) algorithm to detect thermal anomalies, a supervised learning model recognized for its proficiency in both classification and regression tasks (Heller et al., 2003; Ergen & Kozat, 2020; scikit learn). This approach entails adopting an infrared camera as the edge-sensing component to generate thermal profiles of electronic circuits. Simultaneously, an FPGA serves as the edge-computing system, facilitating onboard data processing.

The edge-sensing component responsible for thermal imaging is the FLIR high-resolution LWIR micro-thermal camera module Lepton 3.5 (Systems, 2018). This Original Equipment Manufacturer (OEM) radiometric LWIR camera module features a 160×120 array of active pixels, exhibiting a thermal sensitivity of less than 50 mK. It incorporates an uncooled focal plane array with a pixel size of 12 μm and provides calibrated radiometric output across the entire $19,200$ pixel array.

Integrated into the payload, the Lepton 3.5 functions as an infrared sensor dedicated to thermal imaging. This radiometric sensor ensures accurate, calibrated, and non-contact temperature data across all image pixels in the captured images. Notably, the Lepton 3.5 operates with low power consumption, typically requiring 150 mW and reaching 650 mW during a shutter event. Operating as an uncooled thermal imager, it features a 56° lens and includes a built-in shutter. The device exhibits rapid time-to-image capture, achieving a duration of less than 0.5 seconds, and enables radiometry to map temperatures for every pixel in an image.

3. Methodology

The thermal anomaly detection process on the payload unfolds through distinct stages:

1. Data Collection: This initial phase encompasses the acquisition, gathering, and pre-processing of data. The emphasis lies in collecting data and conducting thermal imaging

utilizing an infrared camera as the edge-sensing component, specifically directed towards electronic circuits.

2. Machine Learning Training and Modeling: In this subsequent stage, data undergoes storage and transformation to train the machine learning algorithm. The training dataset is employed for data ingestion, with a focus on the training process and the computation of optimized parameters.
3. Run-time Inference: Following the training phase, the machine learning model is deployed for run-time data processing. Having been optimized and ready to use during training, the emphasis now shifts to data processing, with the system output comprising the classification labels generated by the machine learning model.

3.1. Data Collection

The establishment of a reference temperature during normal system operation serves as a benchmark for comparing heat generation and distribution within a specific component at any given instance. Through a comparative analysis between thermal distribution profiles and reference profiles, the identification of thermal anomalies, which could potentially lead to system failures, becomes feasible. This proactive approach enables preventive and predictive maintenance by pinpointing hotspots, areas experiencing an increase in temperature.

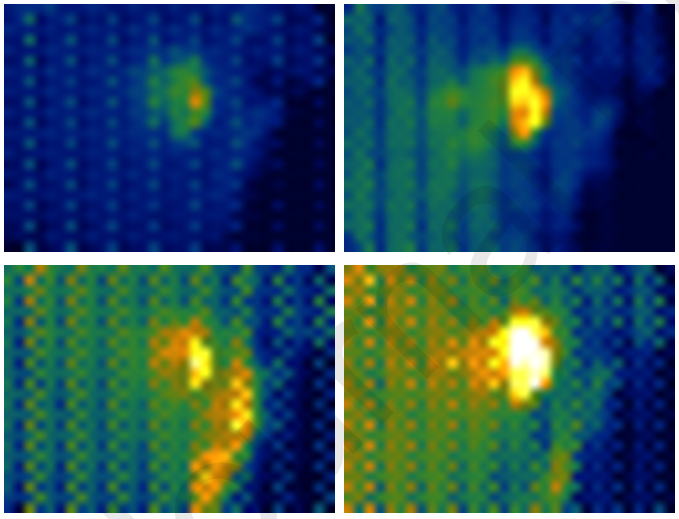


Fig. 2: Thermal images of the testbed at ambient temperature in the laboratory for the low operating power mode (top left) and the high operating power mode for 15 min (top right), 45 min (bottom left) and 60 min (bottom right).

Figure 2 presents thermal images of the laboratory testbed under ambient temperature conditions, captured by the Lepton 3.5 IR thermal camera during both low and high power operating modes. These images, taken over durations of 15 minutes, 45 minutes, and 60 minutes, illustrate the evolution of heat distribution within the electronics as computational tasks are executed. Notably, strategically placed Arduino Uno components and heaters contribute to the observed thermal patterns.

In our study, image generation and data collection were conducted within a controlled laboratory environment. Thermal anomalies were simulated using intensive computational tasks to increase temperatures within the onboard computer, resulting in the creation of a 4-class dataset. Controlled heating elements were also employed to further elevate temperatures in a controlled manner, thereby stressing the satellite payload. Throughout the experiment, parameters such as ambient temperature and humidity, camera calibration settings, and potential sources of noise and interference affecting the image acquisition process were monitored.

3.2. Machine Learning Training and Modeling

The support vector machine (SVM) is a supervised learning technique that allows classifying an image into classes where each element must fit into one class (Vapnik, 2000; Chang & Lin, 2007). Binary classification allows fitting images into two classes by labeling the elements at $\{+1, -1\}$ to map the inputs to the outputs and predict the output based on the mapping.

The SVM algorithm, by definition, searches for the smallest distance between the data points and the decision boundary to make it as large as possible. (Manning et al., 2008; Zhu, 2010)

Primal problem:

- **First statement:** *define a decision boundary that is maximally far away from any data point*

where the margin of the classifier is given by the distance from the decision boundary to the nearest data point. At this point, it can be mentioned that each data point is considered a support vector in the given vector space, i.e. as a vector between the origin and a single data point.

A decision hyperplane is defined by an intercept term b and a normal vector $\vec{w}$, usually referred to as a weight vector, that is perpendicular to the hyperplane. Thus, due to the property of perpendicularity that exists between the hyperplane and the normal vector, all the data points in the hyperplane $\vec{x}$ satisfy

$$\vec{w}^T \cdot \phi(\vec{x}) + b = 0 \tag{1}$$

The training dataset $D = \{(\vec{x}_i, y_i)\}$ is composed of data points $\vec{x}_i \in \mathbb{R}^p$, $i = 1, \ldots, n$ and labels $\vec{y} \in \mathbb{R}^n$ associated with it, where, for binary classification, the decision boundary provides two classes that must fit for a given dataset. Therefore, labels can take the values $y_i \in \{+1, -1\}$ and it can be seen that for binary classification the linear classifier and the definition of functional margin are given by

$$\text{sgn}\,(\vec{w}^T \cdot \phi(\vec{x}) + b) \tag{2}$$

The objective is to find $\vec{w} \in \mathbb{R}^p$ and $b \in \mathbb{R}$ such that the predicted values are given by eq. (2). The functional margin of the i^{th} data point for a given data point $\vec{x}_i$ with respect to the hyperplane $\langle \vec{w}, b \rangle$ is

$$y_i(\vec{w}^T \cdot \phi(\vec{x}_i) + b) \tag{3}$$

The Euclidean distance r_i from a data point $\vec{x}_i$ to the decision boundary is, by definition, perpendicular to the plane and parallel to $\vec{w}$, where, $\vec{w}/\|\vec{w}\|$ is a unit vector in that direction. Thus,

is given by the vector $r_i\vec{w}/\|\vec{w}\|$

$$r_i = y_i(\vec{w}^T \cdot \phi(\vec{x_i}) + b) \cdot \frac{1}{\|\vec{w}\|} \tag{4}$$

The geometrical margin of the classifier is the maximum width of two data points of the two classes, which can be also expressed as twice the minimum width over all data points, i.e. over all the support vectors. Then, by imposing the constraint that the geometric margin is the same as the functional margin, the functional margin will be at least 1 for all data points and equal to 1 for at least one data point.

$$y_i(\vec{w}^T \cdot \phi(\vec{x_i}) + b) \geq 1,\ i = 1, \ldots, n \tag{5}$$

where, the geometric margin ρ is given by

$$\rho = \frac{1}{\|\vec{w}\|} \tag{6}$$

* **Second statement:** *maximize the geometric margin → find* $\{\vec{w}, b\}$ *where* ρ *is maximized* $\forall(\vec{x_i}, y_i) \in D,\ y_i(\vec{w}^T \cdot \phi(\vec{x_i}) + b) \geq 1$

$$\begin{aligned} &\text{maximize}_{\vec{w},b}\ \frac{1}{\|\vec{w}\|} \\ &\text{subject to } y_i(\vec{w}^T \cdot \phi(\vec{x_i}) + b) \geq 1,\ i = 1, \ldots, n \end{aligned} \tag{7}$$

However, maximising $1/\|\vec{w}\|$ is equal to minimizing $\|\vec{w}\|^2$ so the 2nd statement remains a minimization problem.

* **Second statement redefined:** *maximize the geometric margin → find* $\{\vec{w}, b\}$ *where* $\|\vec{w}\|^2$ *is minimized* $\forall(\vec{x_i}, y_i) \in D,\ y_i(\vec{w}^T \cdot \phi(\vec{x_i}) + b) \geq 1$

$$\begin{aligned} &\text{minimize}_{\vec{w},b}\ \|\vec{w}\|^2 = \text{minimize}_{\vec{w},b}\ \vec{w}^T \cdot w \\ &\text{subject to } y_i(\vec{w}^T \cdot \phi(\vec{x_i}) + b) \geq 1,\ i = 1, \ldots, n \end{aligned} \tag{8}$$

where the optimization is given by a quadratic function subject to linear constraints. It can be solved by using standard quadratic programming libraries or by using the Lagrange multipliers α_i in a dual optimization problem where each multiplier is associated with the constraint $y_i(\vec{w}^T \cdot \phi(\vec{x_i}) + b) \geq 1$ in a primal optimization problem.

This is under an ideal scenario where the prediction is perfect. To take into account the real scenario where the dataset cannot be perfectly separable with a hyperplane, the parameter ξ_i refers to the allowed distance to the sample from its correct margin boundary. The penalty term C controls the strength of this penalty and acts as an inverse regularization parameter, meanwhile, $\phi(\vec{x_i})$ maps $\vec{x_i}$ into a higher-dimensional space. Thus, the primal problem is given by

$$\begin{aligned} &\text{minimize}_{\vec{w},b,\xi_i}\ \frac{1}{2}\vec{w}^T \cdot \vec{w} + C\sum_{i=1}^{n}\xi_i \\ &\text{subject to } y_i(\vec{w}^T \cdot \phi(\vec{x_i}) + b) \geq 1 - \xi_i, \\ &\xi_i \geq 0,\ i = 1, \ldots, n \end{aligned} \tag{9}$$

Dual problem:

* **Optimization problem:** *find Lagrange multipliers* $\{\alpha_1, \alpha_2, \ldots \alpha_n\}$ *where* $L(\alpha)$ *is minimized*

Lagrange multipliers α_i is a method to find the local maxima/minima of a function $f(x)$ subject to equality restrictions $g(x)$, i.e. subject to conditions that one or more equations have to be satisfied and convert the restricted problem to a function where the derivative of an unrestricted problem can be applied

$$L(x, \alpha) = f(x) + \sum_i \alpha_i \cdot g_i(x) \tag{10}$$

Hereby, we want to minimize $L(\alpha)$

$$\begin{aligned} &\text{minimize}_{\vec{\alpha}}\ \frac{1}{2}\vec{\alpha}^T Q\vec{\alpha} - \vec{e}^T\vec{\alpha} \\ &\text{subject to } \vec{y}^T\vec{\alpha} = 0,\ 0 \leq \alpha_i \leq C,\ i = 1, \ldots, n \end{aligned} \tag{11}$$

where, the parameter $\vec{e} = [1, \ldots, 1]^T$ is a vector of all ones, Q is a $n \times n$ positive semi-definite matrix $Q_{ij} \equiv y_i y_j \cdot K(\vec{x_i}, \vec{x_j})$ and $K(\vec{x_i}, \vec{x_j}) = \phi(\vec{x_i})^T \cdot \phi(\vec{x_j})$ is the kernel function. The parameter α_i is often called dual coefficients and is upper-bounded by C. Hence, the equation refers to the scenario where the training vectors are implicitly mapped to a higher dimensional space by the function ϕ, which is known as the kernel trick. The solution to this optimization problem, i.e. the decision boundary, for a given data point $\vec{x_j}$ is given by the following function

$$\text{sgn}\left(\sum_{i=1}^{n} y_i\alpha_i \cdot K(\vec{x_i}, \vec{x_j}) + b\right) \tag{12}$$

Finally, the solution to the optimization problem is given by the solution of $\vec{w}$ and b

$$\begin{aligned} \vec{w} &= \sum_{i=1}^{n} y_i\alpha_i\phi(\vec{x_i}) \\ b &= y_i - \vec{w}^T\phi(\vec{x_i}) \quad \forall\vec{x_i}, \alpha_i \neq 0 \end{aligned} \tag{13}$$

where all the non-zero Lagrange multipliers α_i designate that $\vec{x_i}$ is a support vector because they define the decision boundary where Eq. (12) is given by the kernel function

$$\begin{aligned} \text{Linear: } K(x_i, x_j) &= x_i x_j T \\ \text{Polynomial: } K(x_i, x_j) &= (\gamma x_i x_j T + r)d \\ \text{Radial basis function: } K(x_i, x_j) &= e^{(-2\gamma\|x_i - x_j\|)} \\ \text{Sigmoid: } K(x_i, x_j) &= \tanh(\gamma x_i x_j T + r) \end{aligned}$$

where, γ is a term in the kernel parameter for all kernel types except linear and $\gamma > 0$ for radial basis function and polynomial, d is the polynomial degree term in the kernel function for the polynomial kernel, r is the bias term in the kernel function for the polynomial and sigmoid kernels.

The algorithm 1 delineates the pseudo-code governing the training phase of the support vector machine. The input arguments for this phase include the training dataset and the regularization parameter. Within the training phase, computation of

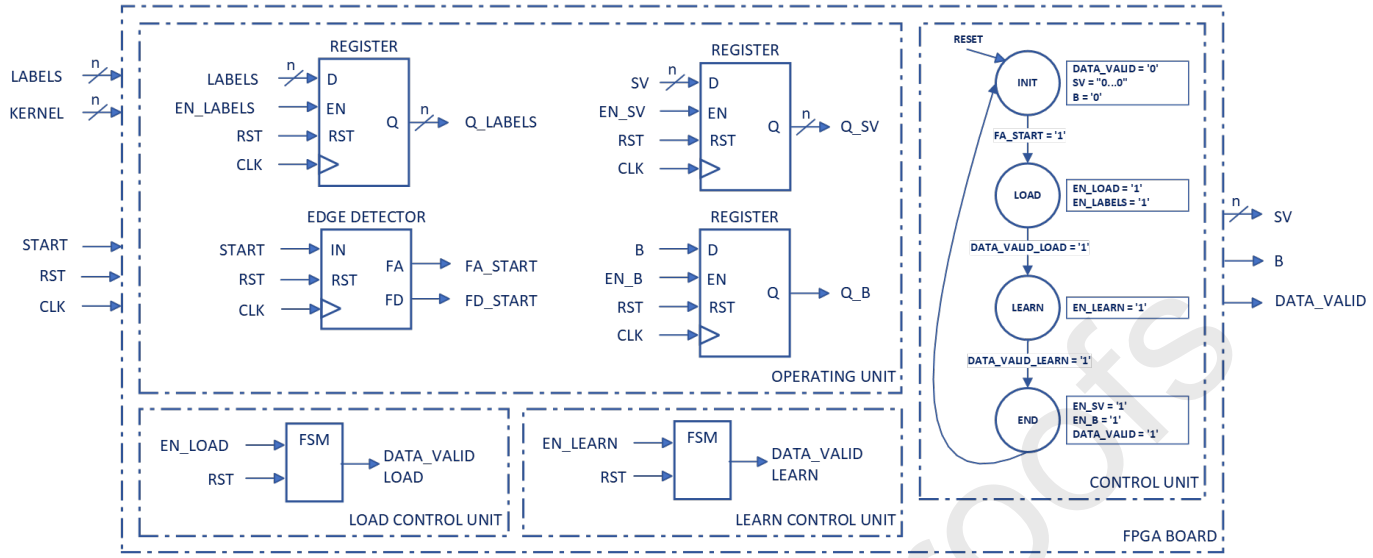


Fig. 3: SVM training digital architecture including three control units and one operating unit.

Algorithm 1: SVM training phase

Input: $D = \{(\vec{x_i}, y_i)\}^n$: training data
C : regularization parameter
Output: $\alpha_i > 0$: support vectors

procedure *svm_training():*
  for $i = 1 : n$ **do**
    find α_i ▷ `Lagrange multipliers`
  end
  return $\alpha_i > 0$ ▷ `Support vectors`
end

the Lagrange multipliers is carried out according to the aforementioned mathematical theory, facilitating the determination of support vectors. These support vectors, serving as the output parameters of the training algorithm, will be the input parameters for the inference algorithm.

Figure 3 illustrates the digital architecture conceived for SVM training, intended for implementation on an FPGA. The high-level design adopts a modular structure featuring an operational unit with a combinational part and logic circuits, accompanied by a three-control unit governed by a finite state machine (FSM). A global asynchronous *reset* is uniformly applicable to combinational blocks, facilitating the systematic clearing of digital components and ensuring a return to a predefined state, as each component signal is interconnected with the higher-level *reset* signal on the board. The sequential operation of the digital system is as follows:

1. Initialization: The activation of the *start* signal announces the entering into the training phase for optimizing the machine learning algorithm. Upon detection of a rising edge in the *start* signal by the edge detector component, the control unit transitions from the *init* to the *load* state.
2. Loading of Labels and Kernel Matrix: In the *load* state, the system is responsible for loading the parameters. A D-type register is employed for the *labels*, while an array of RAM memories handles the *kernel matrix*, with each memory cell storing a row. The completion of this loading process is indicated by the *data_valid_load* signal from *fsm_load*, prompting a transition from the *load* to the *learn* state.
3. Learning and Training: The *learn* state involves the execution of computational tasks for training the SVM classification model. The resolution of this learning process is communicated through the *data_valid_learn* signal from *fsm_learn*, leading to a subsequent transition from the *learn* to the *end* state.
4. Finalization: Subsequently, with the algorithm successfully trained, the support vectors and intercept term become ready to use. The system signals the readiness of the data through the *data_valid* signal, and the values are stored using two D-type registers.

FPGA-based architectures are selected primarily due to their parallel processing capabilities and low-latency execution, which efficiently handle large-scale data processing tasks. Moreover, their inherent flexibility and reconfigurability are advantageous features. The feasibility of onboard labeling depends on several factors, including the availability of reliable ground truth data and the effectiveness of onboard machine learning algorithms for data processing and analysis. In our study, while initial machine learning training occurs on Earth, we incorporate provisions for in-orbit training. This flexibility accommodates potential discrepancies between Earth-based and space-collected data, particularly crucial for maintaining accurate thermal anomaly detection performance. By enabling real-time adjustments and reconfiguration of both FPGA hardware and SVM models, our approach ensures system resilience and adaptability across varying data environments.

3.3. Run-time Inference

The run-time inference phase unfolds subsequent to the training process, where real data is injected into the machine

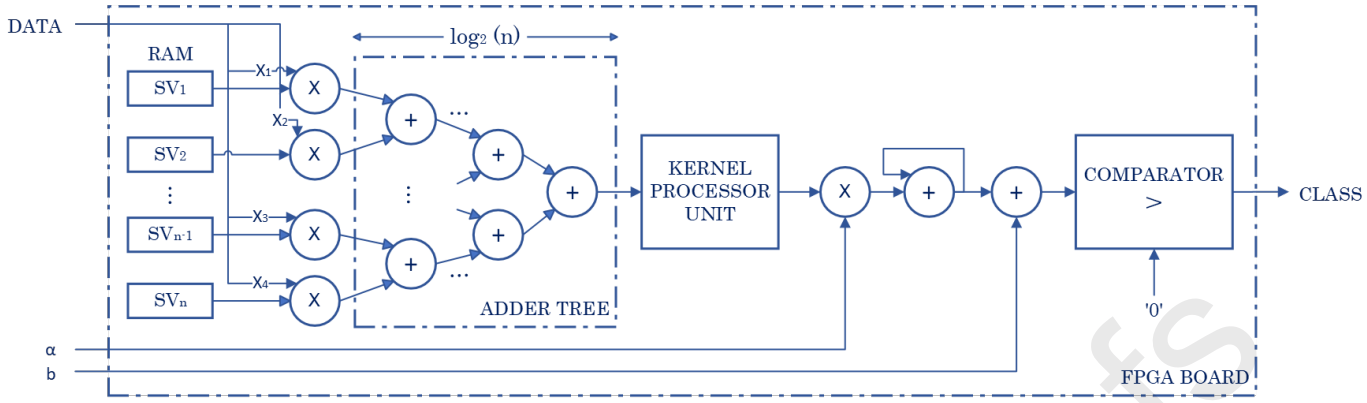


Fig. 4: SVM run-time digital architecture.

learning model to generate accurate predictions and classifications. At this juncture, the model yields a reliable output as it has undergone optimization, given by the learning rate established during the training phase.

Algorithm 2: SVM Classification

Input: $\vec{X} = [X_1, \ldots, X_n]$: input data
$\vec{\alpha} = [\alpha_1, \ldots, \alpha_n]$: support vectors
b : intercept term
Output: $\vec{l} = [l_1, \ldots, l_n]$: classification labels

procedure *svm_classification():*
  for $i = 1 : n$ **do**
    compute $K(\vec{x_i}, \vec{x_i})$ ▷ Kernel
    $l_i = \text{sgn}\left(\sum_{i=1}^{n} y_i \alpha_i \cdot K(\vec{x_i}, \vec{x_j}) + b\right)$
  end
  return $\vec{l}$ ▷ Classification labels
end

Algorithm 2 outlines the pseudo-code governing the run-time inference process within the classification phase of the SVM. It requires essential input parameters such as the input data matrix $\vec{X}$, the vector of Lagrange multipliers $\vec{\alpha}$ (representing support vectors), and the intercept term b. The algorithm systematically computes the kernel function $K(\vec{x_i}, \vec{x_i})$ and, based on the calculated values, assigns classification labels $\vec{l}$ to the input data.

Figure 4 illustrates the digital architecture designed for SVM run-time inference, tailored for FPGA implementation. This architecture handles the computation of the decision boundary, as defined by equation (12). The support vectors, having undergone optimization during the training phase, are stored in Random Access Memory (RAM). Subsequently, the kernel processor unit executes the kernel function, and the combinational blocks process the input data to generate the final decision.

3.4. FPGA resource utilization

The resource utilization of a 4-class SVM classification model implemented on the FPGA Mars XU3 is detailed in Table 1 in terms of Look-Up Tables (LUTs), Flip-Flops (FFs), Digital Signal Processor (DSPs), and Block RAM (BRAM). The FPGA Mars XU3 offers substantial resources, including 476,160 LUTs, 952,320 FFs, 900 DSPs, and 390 BRAM blocks, making it suitable for complex computational tasks. During the SVM training process, the resource utilization is as follows: 7.14% for LUTs, 4.31% for FFs, 10.56% for DSPs, 37.18% for BRAM, with a maximum clock frequency of 200 MHz. This reflects the computational complexity and storage demands during the training phase, as the model learns from the input data and requires storage for both training data and intermediate results.

Table 1: Summary of resource utilization: Look-Up Tables (LUTs), Flip-Flops (FFs), Digital Signal Processor (DSPs), and Block RAM (BRAM) for a 4-class SVM classification model implemented on the Mars XU3 FPGA.

Target \ Resources	LUTs	FFs	DSPs	BRAM
FPGA Mars XU3 resources	476160	952320	900	390
SVM training	35400 (7.43%)	41000 (4.31%)	95 (10.56%)	145 (37.18%)
SVM inference	18400 (3.86%)	23600 (2.48%)	45 (5.00%)	70 (17.95%)

In contrast, the resource utilization for the SVM inference process is significantly lower, as inference primarily involves applying the learned model to new data, which demands fewer computations. Specifically, resource utilization during inference is 3.86% for LUTs, 2.48% for FFs, 5% for DSPs, and 17.95% for BRAM. This underscores the distinct resource demands of the training and inference phases. Training is notably more resource-intensive, particularly regarding LUTs, DSPs, and BRAMs, due to the extensive computation and storage requirements of the learning phase. Conversely, the inference phase, while still resource-demanding, utilizes fewer resources as it focuses on the application of the trained model to new data.

4. Results

This section details the outcomes stemming from the SVM machine learning classification model. The discussion encompasses various facets such as dataset structure, data processing, feature extraction, and an exhaustive evaluation of model performance, including the computation of pertinent performance

metrics. A critical preliminary step involves data preparation, where a set of procedures are defined to format the dataset suitably for analysis by the machine learning algorithm.

Conversely, the pivotal stages of data reduction and attribute sampling play a decisive role in identifying essential features that contribute significantly to forecasting. This process aims to enhance prediction accuracy, providing a more precise estimation of the Region of Interest (ROI). Moreover, data reduction strives to improve dataset quality by diminishing dimensions and mitigating the impact of overweight values.

Lastly, data processing becomes instrumental in segmenting data to delineate one or more ROIs. Subsequently, feature extraction is executed, culminating in the classification of images. Key components in this process include pattern recognition, feature selection, and feature extraction, all crucial in accurately predicting and categorizing the test dataset into predefined classes: normal, heated, overheated, or burnt.

4.1. Structure of the dataset

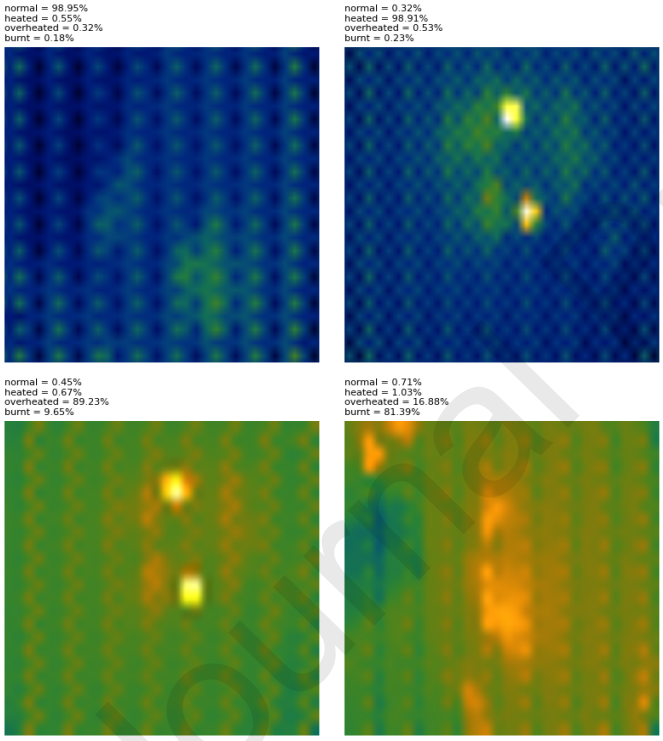


Fig. 5: Thermal images: normal (top left), heated (top right), overheated (bottom left) and burnt (bottom right).

The dataset comprises a total of 1137 thermal images categorized into four classes: normal, heated, overheated, or burnt, denoted by the labels 0, 1, 2, and 3, respectively. Specifically, 536 images belong to the normal class, 527 to heated, 60 to overheated, and 14 to burnt. Figure 5 showcases thermal images belonging to different categories, with clear distinctions in their characteristics. For instance, in the normal category (top left), the temperature is evenly distributed throughout the payload. In contrast, the overheated category (bottom left) exhibits temperature increases in specific regions of the payload.

The dataset, is represented as a $32 \times 24 \times 3$ array, where 32 represents the width and 24 the height of the image for 3 channels, i.e. one for each RGB color, which takes values between $[0, 255]$ depending on the intensity of each pixel. Subsequently, each pixel is designated as a feature and normalized to a range between $[-1, 1]$. The Albumentations library is utilized for data augmentation, applying the following 9 transformations: RGBShift, ChannelShuffle, RandomBrightnessContrast, Blur, MedianBlur, ToGray, ImageCompression, VerticalFlip, and HorizontalFlip. Figure 6 depicts the 9 transformations applied to the original images. Each row represents a different category: normal, heated, overheated, or burnt, while each column corresponds to a specific transformation applied to the images of that category. In particular, the 3rd column, which shows the ChannelShuffle transformation, exhibits a pronounced color shift, particularly for images in the overheated category (3rd row), making them appear as if they belong to the burnt category. Similarly, images in the heated category (2nd row) appear to be from the overheated category. These data augmentations are intended to enhance the robustness of the machine learning model by introducing variability into the training data, thereby improving the model generalization capabilities. As a result, the dataset is transformed into a structured Pandas DataFrame with x samples and y features $(x, y) : (11370, 2304)$ for the data augmentation dataset and $(x, y) : (1137, 2304)$ for the original dataset, where each row corresponds to a sample, and each column represents a feature.

Subsequently, the dataset is partitioned into distinct training and testing sets, with 70% of the samples allocated for training the SVM machine learning classification model, while the remaining 30% are reserved for testing and model evaluation purposes. Consequently, the training and testing datasets for the data augmentation dataset exhibit dimensions of $(X_{\text{train}}, y) : (7959, 2304)$ and $(X_{\text{test}}, y) : (3411, 2304)$, respectively. For the original dataset, the dimensions are $(X_{\text{train}}, y) : (795, 2304)$ and $(X_{\text{test}}, y) : (342, 2304)$.

4.2. Data processing and feature extraction

Effective feature selection and extraction are paramount for accurate predictions, especially when dealing with a substantial number of features. The Histogram of Oriented Gradients (HOG) diagram serves as a valuable tool for selecting representative features from the data (Li & Su, 2015; Al-Rababah et al., 2021). Functioning as a texture analysis method, HOG segregates the data into cells and computes histograms based on gradient directions. Subsequently, block normalization is employed, encompassing a defined number of cells, to assess and standardize histograms at the cell level within each block. The HOG parameters encompass 8×8 pixels per cell, 2×2 cells per block, and L2-Hys normalization utilizing a Euclidean norm with a hysteresis factor of 0.2, as depicted in Figure 8.

The HOG diagram serves as a tool for extracting meaningful features for thermal anomaly detection. These features are applied within the SVM algorithm, functioning as a 4-class classification model. Figure 7 illustrates the thermal images after the computation of the HOG diagram. This representation under-

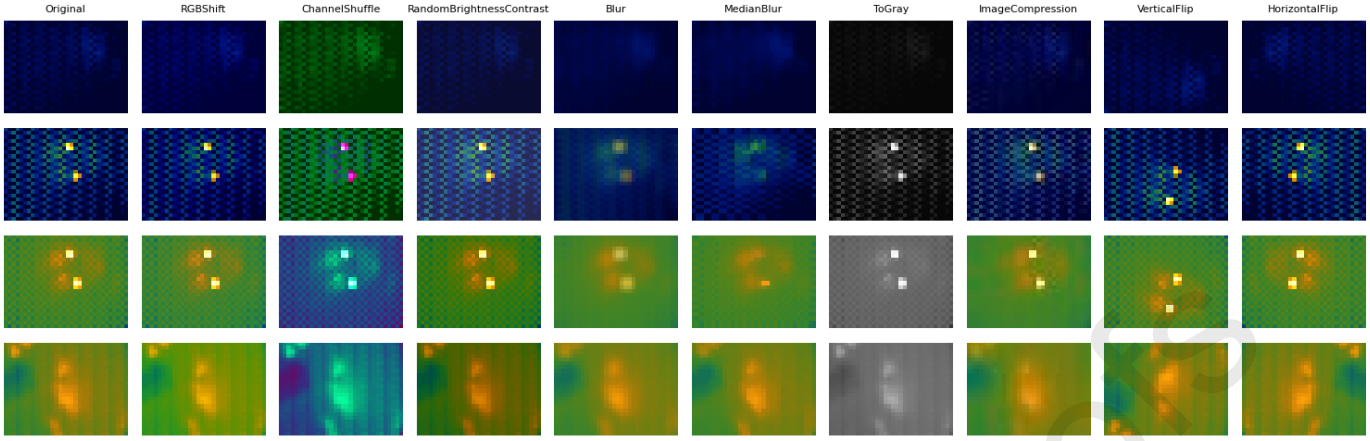


Fig. 6: Data augmentation techniques: RGBShift, ChannelShuffle, RandomBrightnessContrast, Blur, MedianBlur, ToGray, ImageCompression, VerticalFlip, and HorizontalFlip, implemented using the Albumentations library.

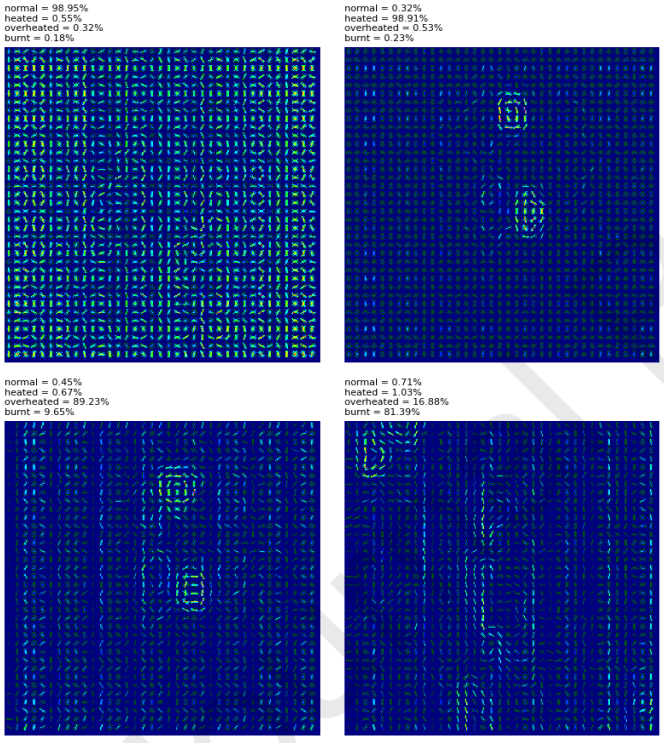


Fig. 7: HOG diagram: normal (top left), heated (top right), overheated (bottom left) and burnt (bottom right).

scores the uniform distribution of gradients across the payload in the normal category (top left). Conversely, the overheated category (bottom left) exhibits concentrated gradients in specific payload regions, indicative of heightened intensity. These areas correspond to hot spots, reflecting a substantial increase in temperature.

Implementing HOG feature selection and extraction on an FPGA presents challenges due to its iterative and intensive computation requirements. Therefore, we opted to execute these tasks on the onboard CPU. HOG involves computing gradients and histogram calculations over image patches, requiring frequent memory accesses and complex arithmetic operations. Although FPGA platforms provide parallel processing capabilities, efficiently handling the memory bandwidth and computational demands of HOG may surpass the FPGA resources allocated for logic and signal processing tasks.

The computational demands of SVM, particularly when applied to high-dimensional data and non-linear kernels, are significantly higher compared to HOG feature extraction method, underscoring the challenges of scaling SVM for large datasets (Bottou, 2010). The computational complexity of HOG is generally linear relative to the number of pixels in an image, involving gradient computation and histogram updates for each pixel (Dalal & Triggs, 2005). Research on HOG implementation in FPGAs focuses on optimizing the computation of gradient orientations, cell histograms, and block normalization to achieve real-time performance. Although these operations are data-intensive, they mainly involve basic arithmetic and aggregation, making them less demanding on computational resources. Conversely, SVM is inherently more computationally demanding due to the need to solve an optimization problem, particularly with large datasets and complex kernel functions (Bottou & Lin, 2007; Keerthi & Lin, 2003). The complexity of SVM can be quadratic or cubic relative to the number of data points due to the need to compute the kernel matrix and solve the optimization problem (Tsang et al., 2005; Cervantes et al., 2008), which results in higher utilization of FPGA resources such as logic units and memory.

4.3. Model construction and training

The SVM algorithm, treating each pixel as an individual feature, leads to a dataset with 2304 features. To address this high-dimensional feature space, a pre-processing stage utilizing Principal Component Analysis (PCA) is implemented to reduce dimensionality, selecting the number of components required to capture 85% of the explained variance that is then fed into the SVM classifier within a single pipeline. The optimal parameters for establishing the most effective SVM classification model are determined through the implementation of k-fold

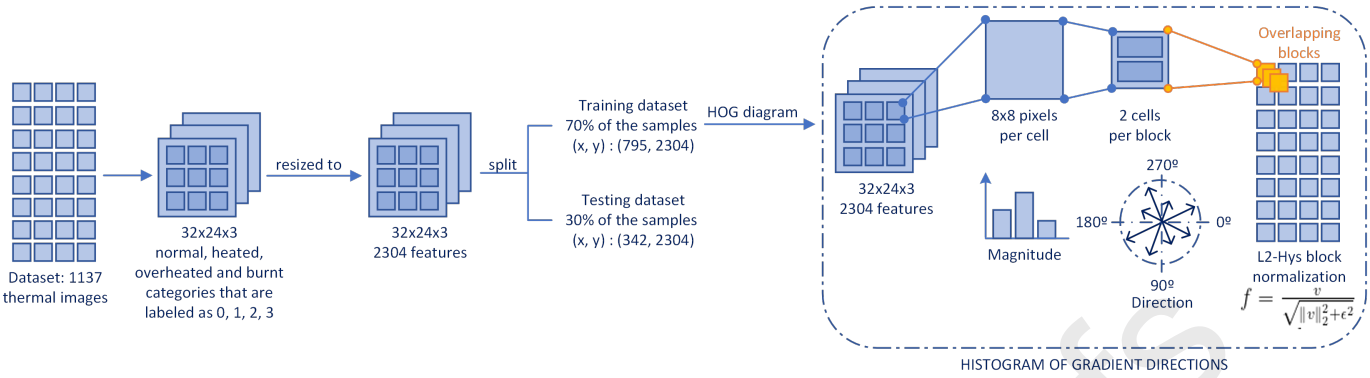


Fig. 8: Data processing and feature extraction.

cross-validation. The hyper-parameters subjected to tuning for achieving the optimal model encompass:

* The kernel function, an integral component embedded in the kernel of the machine learning classification model, can adopt different functions such as linear, polynomial, or Radial Basis Function (RBF). Typically, polynomial and RBF are preferred when dealing non-linear hyper-planes.

* The regularization or penalty parameter, denoted as C, serves as a metric for the error rate within the machine learning model. It dictates the acceptable level of error during the optimization process, thereby influencing the trade-off between the decision boundary and the misclassification term. Lower values of C result in a hyperplane with a smaller margin, while higher values indicate a hyperplane with a larger margin.

* The gamma parameter significantly influences the curvature of the decision boundary. It represents the inverse of the radius of influence of the samples chosen as support vectors by the model during the training phase. Elevated gamma values lead to a decision boundary with increased curvature, whereas lower values result in a decision boundary with reduced curvature.

In our study, we examined four configurations: SVM, HOG + SVM, PCA + SVM, and HOG + PCA + SVM to identify the most suitable model for accurately classifying thermal anomalies in a 4-class SVM machine learning model. Each configuration was evaluated on both the original dataset and a version enhanced with data augmentation techniques, enabling us to assess the impact of data augmentation on model performance. The SVM configuration serves as a baseline, while HOG and PCA are used to extract essential features and reduce dimensionality, aiming to enhance model efficiency and accuracy. By exploring these configurations, we seek to determine whether data augmentation, combined with specific feature extraction and reduction methods, can significantly improve the generalization and accuracy of the SVM machine learning model.

The hyperparameters best suited for the SVM classification model (C, kernel) and performance metrics (accuracy, precision, recall, F1-score, MCC), optimized through 5-fold cross-validation for the four configurations on both the original and augmented datasets, are summarized in Table 2. On the original dataset, the SVM configuration achieves perfect performance metrics, suggesting ideal classification with minimal computational cost. However, this is attributed to the dataset insufficient representativeness, particularly for the overheated and burnt minority classes, which contain only 60 and 14 images, respectively, making the evaluation unreliable. To address this limitation, we applied data augmentation techniques, expanding our dataset by a factor of 10 to improve model robustness and achieve a more balanced and reliable training process.

For the augmented dataset, the HOG + SVM configuration stands out by achieving a balance between high performance (accuracy: 0.9549, MCC: 0.9234) and moderate computational cost, positioning it as a robust option for practical applications. In contrast, the PCA + SVM and HOG + PCA + SVM configurations exhibit more pronounced drops in MCC and overall performance. While PCA + SVM yields a lower accuracy of 0.7877, it significantly reduces the number of features from 2304 to 3, compared to the SVM-only configuration. This reduction minimizes dataset size and enhancing computational efficiency, but it comes at the cost of a notable drop in MCC to 0.6267, compared to the HOG + SVM configuration, which achieves an MCC of 0.9234.

Moreover, the HOG + SVM configuration requires less training time, as shown in Table 3, offering a pragmatic trade-off between performance and efficiency. The substantial increase in computational time and energy consumption observed for SVM and PCA + SVM underscores the trade-off between achieving model robustness and maintaining resource efficiency. This highlights the necessity of careful balancing between performance and computational constraints. Additionally, the importance of dataset management and the implementation of data augmentation are emphasized, especially to address the limitations of underrepresented minority classes.

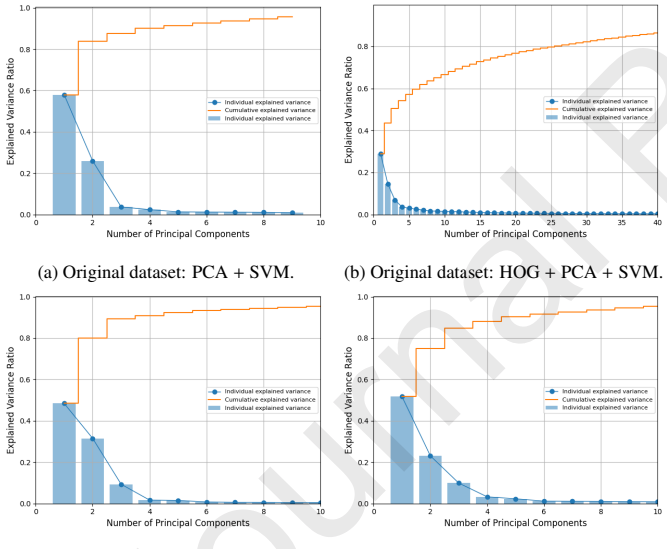
As illustrated in Figure 9, data augmentation proved to be beneficial for the HOG + PCA + SVM configuration, significantly reducing the number of principal components required to explain 85% of the variance. Specifically, with the original dataset, this configuration needed 37 components, whereas, after applying data augmentation, the number of required components decreased to just 5. This indicates a more robust and efficient feature representation in the augmented dataset. In con-

Table 2: Summary of SVM hyperparameters (C, kernel) and performance metrics (accuracy, precision, recall, F1-score, MCC) for various configurations: SVM, HOG + SVM, PCA + SVM and HOG + PCA + SVM. Results are provided for both the original and data augmentation dataset, including training and testing dataset sizes. The optimal configuration for our study highlighted in bold.

Configuration	Dataset	X_train	X_test	C	Kernel	Accuracy (weighted)	Precision (weighted)	Recall (weighted)	F1-score (weighted)	MCC
SVM	Original	(795, 2304)	(342, 2304)	0.001	linear	1	1	1	1	1
HOG + SVM	Original	(795, 216)	(342, 216)	0.01	linear	0.9971	0.9972	0.9971	0.9971	0.9948
PCA + SVM	Original	(795, 4)	(342, 4)	0.01	linear	0.9152	0.9190	0.9152	0.9160	0.8487
HOG + PCA + SVM	Original	(795, 37)	(342, 37)	0.001	linear	0.9942	0.9947	0.9942	0.9943	0.9896
SVM	Augmented	(7959, 2304)	(3411, 2304)	0.001	linear	0.9405	0.9459	0.9405	0.9403	0.8974
HOG + SVM	**Augmented**	**(7959, 216)**	**(3411, 216)**	**0.01**	**linear**	**0.9549**	**0.9706**	**0.9549**	**0.9598**	**0.9234**
PCA + SVM	Augmented	(7959, 3)	(3411, 3)	0.001	linear	0.7877	0.7942	0.7877	0.7878	0.6267
HOG + PCA + SVM	Augmented	(7959, 5)	(3411, 5)	0.001	linear	0.6737	0.7938	0.6737	0.7094	0.5042

Table 3: Summary of the CPU processing time for HOG, PCA, and SVM computation, as well as the prediction time for the testing dataset and energy consumption for various configurations: SVM, HOG + SVM, PCA + SVM and HOG + PCA + SVM. Results are provided for both the original and data augmentation dataset, with the optimal configuration for our study and the total computational time highlighted in bold.

Configuration	Dataset	HOG CPU process time (s)	PCA CPU process time (s)	SVM CPU process time (s)	X_test prediction CPU process time (s)	**Total time (s)**	Energy consumption (J)
SVM	Original	N/A	N/A	0.34	0.03	**0.37**	1.85
HOG + SVM	Original	6.18	N/A	0.17	0.02	**6.37**	31.85
PCA + SVM	Original	N/A	435.35	0.02	0.00	**435.37**	2176.85
HOG + PCA + SVM	Original	6.62	36.22	0.03	0.00	**42.87**	214.35
SVM	Augmented	N/A	N/A	4407.60	41.18	**4448.78**	22243.9
HOG + SVM	**Augmented**	**42.28**	**N/A**	**21.03**	**1.95**	**65.26**	**326.3**
PCA + SVM	Augmented	N/A	1573.03	9.92	0.49	**1583.44**	7917.2
HOG + PCA + SVM	Augmented	38.22	5.15	8.34	0.64	**52.35**	261.75



(a) Original dataset: PCA + SVM. (b) Original dataset: HOG + PCA + SVM.

(c) Data augmentation: PCA + SVM. (d) Data augmentation: HOG + PCA + SVM

Fig. 9: Comparison of scree plots with PCA explained variance and cumulative variance for PCA + SVM and HOG + PCA + SVM configurations.

trast, the PCA + SVM configuration required only 4 components for the original dataset and 3 components for the augmented dataset. Remarkably, the first principal component alone explains over 50% of the variance in both cases, demonstrating that only a small number of components are necessary to capture the majority of the dataset variability. These observations suggest that while data augmentation significantly enhances the feature extraction process for the HOG + PCA + SVM pipeline, the PCA + SVM configuration inherently achieves effective dimensionality reduction with minimal components, independently of dataset augmentation. This underscores the effectiveness of PCA in reducing dimensionality while preserving critical information.

Conversely, the SVM and PCA + SVM configurations proved to be impractical for larger datasets, primarily due to a drastic increase in training time following data augmentation. As discussed earlier, while data augmentation improved feature representation and model robustness, it also introduced considerable computational demands. For the original dataset comprising 1137 thermal images, training durations were manageable: SVM required only 0.37 seconds, and PCA + SVM completed training in 435.37 seconds. However, with the augmented dataset containing 11370 images, training times escalated sharply: SVM required 4448.78 seconds, and PCA + SVM surged to 1583.44 seconds. Such prolonged training durations render these configurations unfeasible for larger datasets, creating a substantial bottleneck during the training phase in space. The HOG + SVM configuration achieves a remarkably low training time. For the original dataset, it required only 6.37 seconds, and even with the augmented dataset, which increased the dataset size by a factor of ten, the training time remained at 65.26 seconds. This represents one of the lowest training times among all tested configurations. Coupled with its high performance (accuracy: 0.9549, MCC: 0.9234), HOG + SVM emerges as the most viable configuration for our study.

Table 4 presents a detailed summary of the cross-validation scores for each fold for the SVM, HOG + SVM, PCA + SVM, and HOG + PCA + SVM configurations, evaluated on both the original and data augmented datasets. The results highlight the reliability of the SVM and HOG + SVM configurations on the original dataset, with an average cross-validation scores of 0.9724 and 0.9431 on the training dataset and 1 and 0.9921 on the testing dataset, respectively. However, as discussed previ-

Table 4: Summary of cross-validation scores for each fold for various configurations: SVM, HOG + SVM, PCA + SVM and HOG + PCA + SVM. Results are provided for both the training and testing datasets, with the optimal configuration for our study and average values highlighted in bold.

Configuration	Dataset	K-folds	Cross-validation score (training dataset)						Cross-validation score (testing dataset)					
			First fold	Second fold	Third fold	Fourth fold	Firth fold	**Average**	First fold	Second fold	Third fold	Fourth fold	Firth fold	**Average**
SVM	Original	5	0.9319	0.9966	0.9333	1	1	**0.9724**	1	1	1	1	1	**1**
HOG + SVM	Original	5	0.9353	0.9449	0.9333	0.9020	1	**0.9431**	1	1	1	0.9603	1	**0.9921**
PCA + SVM	Original	5	0.7955	0.8880	0.8169	0.8818	0.9664	**0.8697**	0.9531	0.9452	0.9766	0.9765	0.9478	**0.9598**
HOG + PCA + SVM	Original	5	0.9205	0.9449	0.9333	1	0.9836	**0.9565**	1	1	1	0.9603	1	**0.9921**
SVM	Augmented	5	0.9342	0.9166	0.8812	0.9191	0.9101	**0.9123**	0.9191	0.8786	0.9161	0.9521	0.9101	**0.9152**
HOG + SVM	**Augmented**	**5**	**0.9011**	**0.8918**	**0.8600**	**0.8926**	**0.8957**	**0.8882**	**0.8858**	**0.8771**	**0.8767**	**0.8764**	**0.8692**	**0.8770**
PCA + SVM	Augmented	5	0.6734	0.6713	0.6693	0.6717	0.6745	**0.6721**	0.6427	0.6452	0.6476	0.6631	0.6452	**0.6487**
HOG + PCA + SVM	Augmented	5	0.4583	0.4413	0.4221	0.4528	0.5083	**0.4566**	0.4311	0.4196	0.4514	0.4476	0.4231	**0.4346**

Table 5: Summary of PCA components and support vectors per class for various configurations: SVM, HOG + SVM, PCA + SVM, and HOG + PCA + SVM. Results are provided for both the original and data augmentation datasets, with the optimal configuration for our study and total number of support vectors highlighted in bold.

Configuration	Dataset	PCA components	Support vectors per class				**Number of support vectors**
			Normal	Heated	Overheated	Burnt	
SVM	Original	N/A	26	32	12	3	**73 (6.42 %)**
HOG + SVM	Original	N/A	27	22	22	4	**75 (6.59 %)**
PCA + SVM	Original	4	70	71	20	5	**166 (14.59 %)**
HOG + PCA + SVM	Original	37	65	66	14	3	**148 (13.01 %)**
SVM	Augmented	N/A	914	579	101	31	**1625 (14.29 %)**
HOG + SVM	**Augmented**	**N/A**	**589**	**797**	**152**	**25**	**1563 (13.74 %)**
PCA + SVM	Augmented	3	2271	2464	345	76	**5156 (45.34 %)**
HOG + PCA + SVM	Augmented	5	2698	3392	400	91	**6581 (57.88 %)**

ously, these ideal scores for the original dataset are indicative of the limited representativeness of the minority classes. This reinforces the earlier point regarding the need for data augmentation to improve model generalization. Upon applying data augmentation, the SVM configuration still achieves strong performance, maintaining an average cross-validation score of 0.9123 on the training dataset and 0.9152 on the testing dataset. The HOG + SVM configuration also performs well with a slightly drop on the testing dataset accuracy to 0.8770. The PCA + SVM and HOG + PCA + SVM configurations exhibit lower scores, reflecting the trade-offs between feature dimensionality reduction and model performance. The PCA + SVM approach, in particular, yields an average testing score of 0.6487.

Table 5 summarizes the PCA components and support vectors for the SVM, HOG + SVM, PCA + SVM, and HOG + PCA + SVM configurations for both the original and augmented datasets. Data augmentation notably increased the number of support vectors required, particularly in the PCA + SVM configuration, exhibiting a significant rise to 5156 support vectors, representing 45.34% of the dataset. Conversely, the HOG + PCA + SVM demanded the highest number of support vectors in the augmented dataset, with 6581 support vectors, accounting for 57.88% of the dataset. This indicates a greater computational resource requirements. The HOG + SVM configuration performed well in both the original and augmented datasets requiring 75 support vectors (representing 6.59% of the dataset), and 1563 support vectors (accounting for 13.74% of the dataset), respectively.

PCA extraction involves matrix operations such as eigenvalue decomposition, which are computationally intensive and memory-dependent. These operations require efficient handling of large data matrix and precise numerical calculations, posing challenges for FPGA architectures optimized for parallelism and low-latency tasks. While FPGA-based solutions excel in certain data processing tasks, the implementation of PCA on FPGA may face limitations in terms of memory bandwidth, scalability, and computational efficiency.

Similarly to HOG feature selection and extraction, implementing PCA for dimensionality reduction and principal component extraction on an FPGA poses challenges due to its computationally intensive matrix operations, such as eigenvalue decomposition. These operations demand efficient handling of large data matrices and numerical computations, thereby requiring their execution on the onboard CPU.

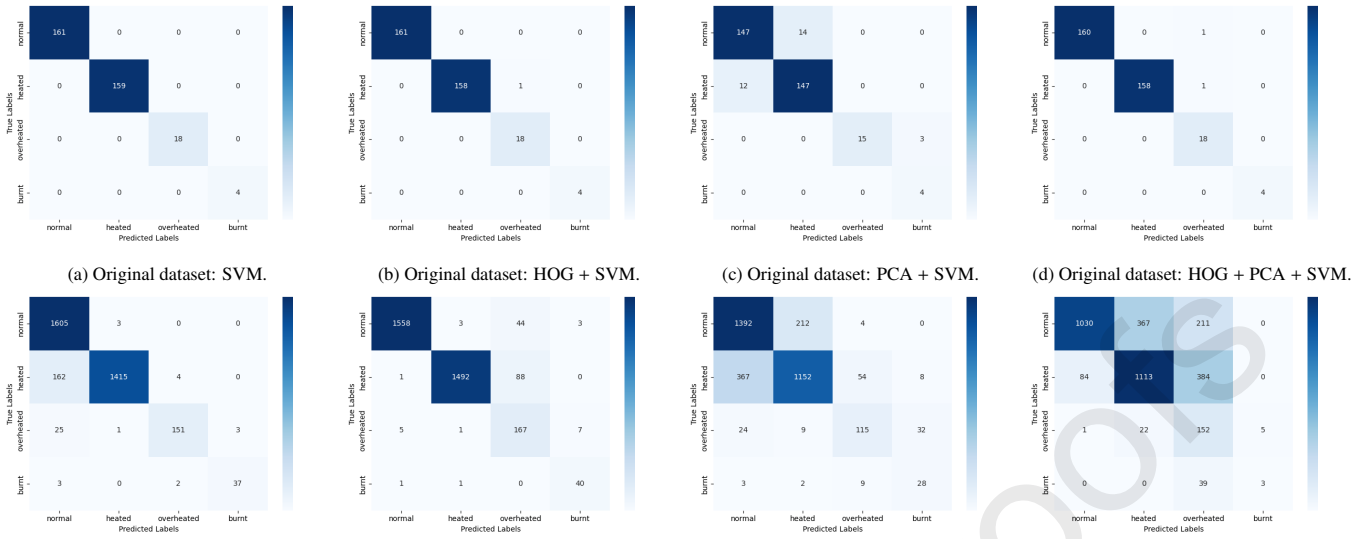
Using a pre-trained model, by definition, removes the need of having labels on-board if the number and type of classes remain the same, as is the case considered in this study involving 4 classes. Only in the event of statistical drift would the model need to be retrained with updated image examples and labels (Ackerman et al., 2022, 2021). However, given the statistical distribution of images for the type of image data described in this paper and the time window during which these images were collected and examined, the data is not expected to undergo any statistical drift. A statistical drift may occur over a sufficiently long-enough period where sensor degradation, for example, would set in.

4.4. Evaluation of the model

This section presents the model evaluation, encompassing an analysis of the confusion matrix, decision boundary, and the Area Under the Curve (AUC) of the Receiver Operating Characteristic (ROC) curve.

4.4.1. Confusion matrix

The confusion matrix, a square matrix of dimensions $N \times N$ where N represents the number of classes, evaluates the performance of an N-class classification model. This is accomplished



(a) Original dataset: SVM. (b) Original dataset: HOG + SVM. (c) Original dataset: PCA + SVM. (d) Original dataset: HOG + PCA + SVM.

(e) Data augmentation: SVM. (f) Data augmentation: HOG + SVM. (g) Data augmentation: PCA + SVM. (h) Data augmentation: HOG + PCA + SVM.

Fig. 10: Comparison of confusion matrix for SVM, HOG + SVM, PCA + SVM, and HOG + PCA + SVM experiments on both the original dataset and with data augmentation.

by comparing actual targets with predictions generated by the machine learning algorithm. Each entry in the matrix quantifies the number of predictions accurately or inaccurately classified by the machine learning classification model. The terminology used for classification outcomes are defined as follows:

* True Positive (TP): Refers to the number of positive predictions that have been correctly classified, signifying that the actual target aligns with the predicted one.

* True Negative (TN): Denotes the number of negative predictions that have been correctly classified, indicating that the actual target corresponds to the predicted one.

* False Positive (FP) or Type I Error: Represents the number of predictions where the negative class is inaccurately predicted as positive, indicating a mismatch between the actual and predicted targets.

* False Negative (FN) or Type II Error: Signifies the number of predictions where the positive class is inaccurately predicted as negative, indicating a disparity between the actual and predicted targets.

Figure 10 depicts the 4×4 confusion matrix associated with the 4-class SVM classification model, offering an overview of the machine learning algorithm performance for SVM, HOG + SVM, PCA + SVM and HOG + PCA + SVM configurations both for the original and data augmentation datasets. Specifically, using the HOG + SVM configuration with the augmented dataset, the SVM classifier accurately predicts the normal category for 1558 thermal images, the heated category for 1492 images, the overheated category for 167 images, and the burnt category for 40 images (as noted along the diagonal in Figure 10f). Notably, the HOG + PCA + SVM configuration with augmented data exhibits substantial misclassification within the heated and overheated classes. At this point, it can be mentioned that in our specific application, when the model misclassifies an overheated image as belonging to the burnt category, the system will still raise an alert due to the elevated temperature. This approach ensures that any potential overheating issue is addressed. By adopting this conservative approach, we prioritize early detection of critical temperature conditions, thereby maintaining the robustness of the system and mitigating the risk of failure, even if the model classifications are not entirely precise. Conversely, the incorrect identification of burnt images, classifying them as overheated poses a significant risk, as it may lead the system to misjudge a critical high-temperature condition as merely overheated.

4.4.2. *Decision boundary*

The decision boundary is defined by a collection of lines and regions designed to differentiate one class from the others. Figure 12 represents the decision boundary of the SVM classifier, providing insight into the discernibility of training and test data samples for SVM, HOG + SVM, PCA + SVM and HOG + PCA + SVM configuration both for the original and data augmentation datasets. Furthermore, the figure delineates four well-defined regions within the feature space corresponding to each of the four classes, achieved through PCA for the purpose of visualizing the decision boundaries in a 2D plot. Each region represents one of the four predefined categories: normal (dark blue), heated (light blue), overheated (light red), and burnt (dark red).

As depicted in the figure, the HOG + SVM and HOG + PCA + SVM models with data augmentation (Figures 12f and 12h), as well as the SVM and PCA + SVM models with the original dataset (Figures 12a and 12c), demonstrate superior class separation by establishing a more effective decision boundary. This configurations yield a clearer distribution, where the 4 classes are distinctly separated and well-distributed across the

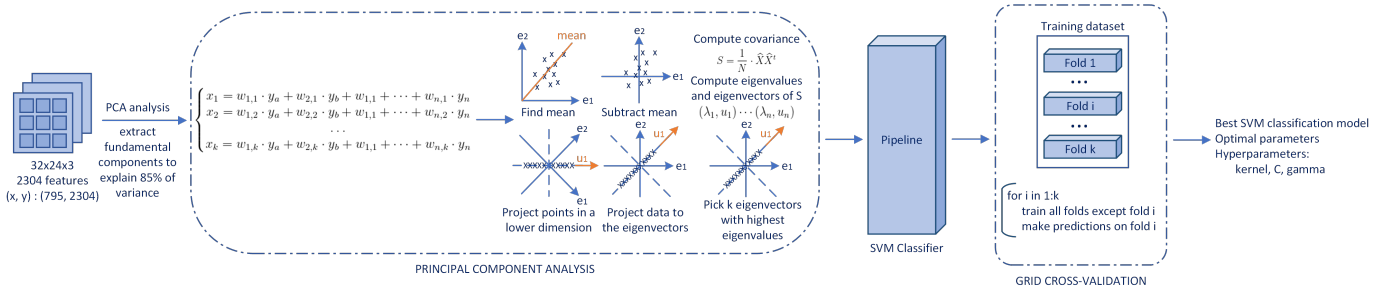
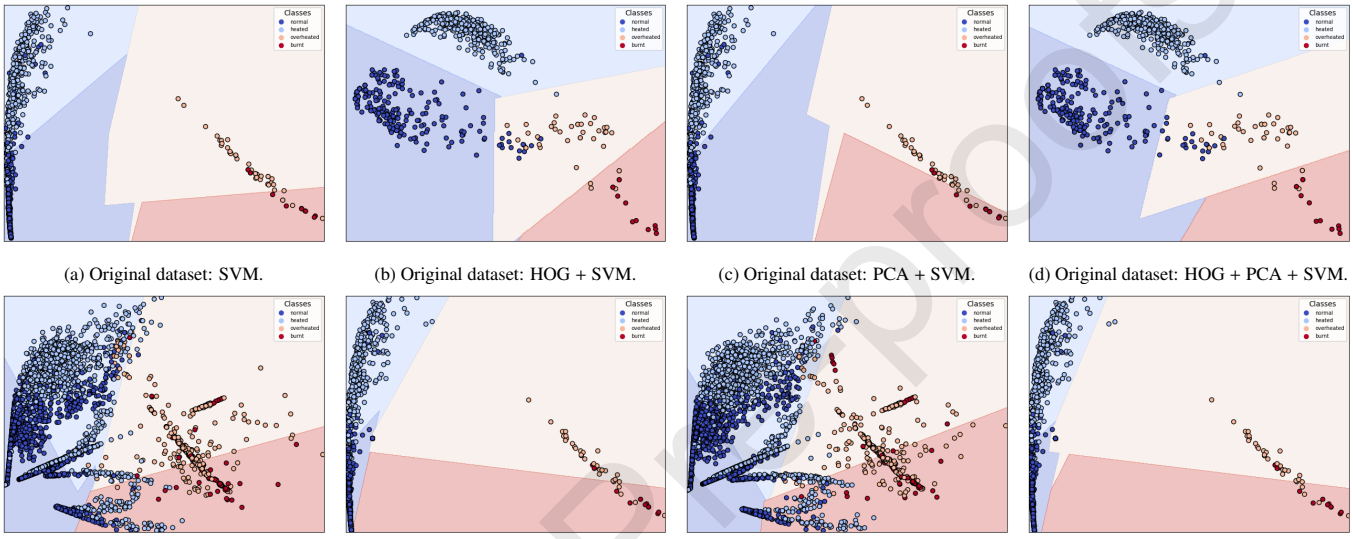


Fig. 11: Model construction and training.



(a) Original dataset: SVM. (b) Original dataset: HOG + SVM. (c) Original dataset: PCA + SVM. (d) Original dataset: HOG + PCA + SVM.

(e) Data augmentation: SVM. (f) Data augmentation: HOG + SVM. (g) Data augmentation: PCA + SVM. (h) Data augmentation: HOG + PCA + SVM.

Fig. 12: Comparison of decision boundaries for SVM, HOG + SVM, PCA + SVM, and HOG + PCA + SVM experiments on both the original dataset and with data augmentation training datasets.

feature space. In contrast, the SVM and PCA + SVM configurations with the augmented dataset present a less defined decision boundary. Surprisingly, the application of data augmentation for both the SVM and PCA + SVM configurations resulted in less-defined decision boundaries. While data augmentation is typically expected to enhance model generalization by increasing the diversity of the training dataset, in this case, it appeared to introduce noise and complexity that affects the decision boundaries. As a consequence, the models struggled to accurately delineate between the various classes of thermal images, leading to increased misclassifications. This unexpected outcome highlights the nuanced impact of data augmentation techniques, suggesting that their effectiveness may vary significantly depending on the model and specific characteristics of the dataset.

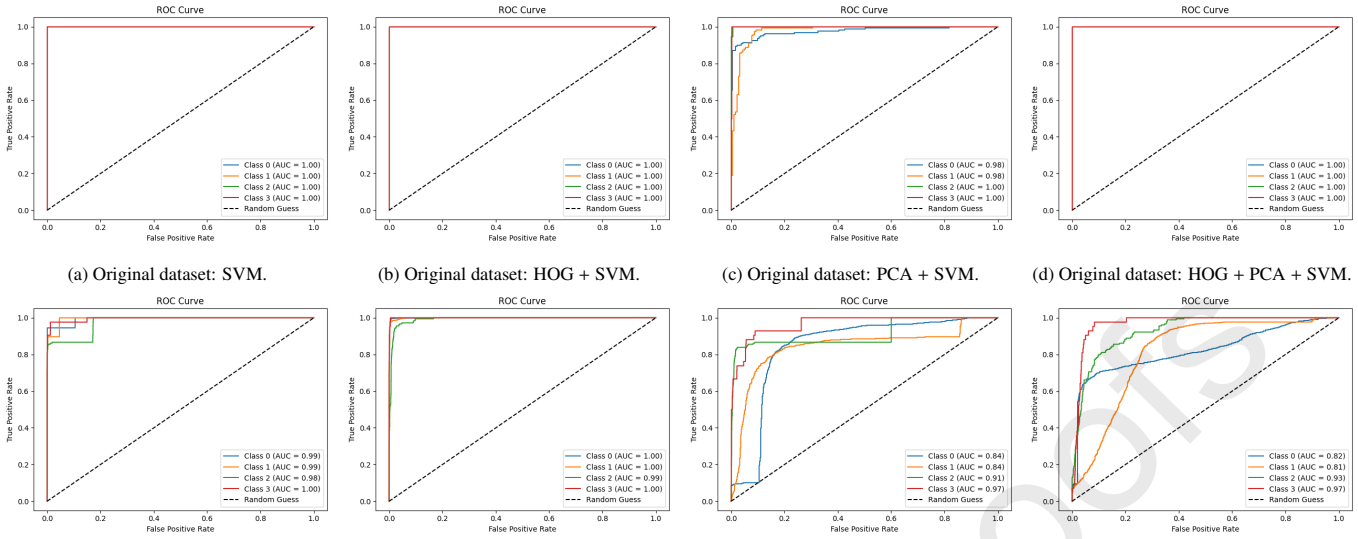
4.4.3. Receiver Operating Characteristics (ROC) curve

Figure 13 presents the ROC curves and corresponding AUC scores for the SVM, HOG + SVM, PCA + SVM, and HOG + PCA + SVM configuration for both the original and data augmentation datasets. For the original dataset, the SVM, HOG + SVM, and HOG + PCA + SVM configurations achieved AUC scores of 1 across all classes, demonstrating excellent classification performance between true positive and false positive rates. The PCA + SVM configuration on the original dataset, however, showed a slight decline in performance, particularly for Class 0 with an AUC of 0.98. When data augmentation was introduced, specifically, the PCA + SVM and HOG + PCA + SVM configurations displayed the most noticeable drops in AUC, with values for certain classes falling below 0.90, reflecting challenges in maintaining classification accuracy on augmented data. The SVM and HOG + SVM configurations showed relatively better resilience to data augmentation, achieving AUC scores close to 1 for most classes, which suggests they are more robust to the variability introduced by augmentation.

5. Discussion

This section recapitulates the primary outcomes stemming from the application of edge computing in the context of a specific case study focused on thermal anomaly detection in space systems, elucidating the associated implications and findings, emphasizing key considerations and uncertainties.

▹ **Controlled Environment Limitations:** The dataset comprises thermal images captured from a payload in a controlled terrestrial environment under ambient conditions. These controlled conditions, while facilitating initial machine learning



(a) Original dataset: SVM. (b) Original dataset: HOG + SVM. (c) Original dataset: PCA + SVM. (d) Original dataset: HOG + PCA + SVM.

(e) Data augmentation: SVM. (f) Data augmentation: HOG + SVM. (g) Data augmentation: PCA + SVM. (h) Data augmentation: HOG + PCA + SVM.

Fig. 13: Comparison of area under the ROC curve for SVM, HOG + SVM, PCA + SVM, and HOG + PCA + SVM experiments on both the original dataset and with data augmentation.

training and modeling on Earth, presents an inherent limitation as it does not faithfully replicate the conditions prevailing in the space environment. The study recognizes this limitation, acknowledging the need for subsequent comparison with results obtained from the thermal vacuum chamber.

▹ **Data Augmentation and Dataset Vulnerabilities:** To tackle the controlled environment constraint, data augmentation techniques were applied to the thermal images in order to increase the amount of data required for statistical analysis, especially for the burnt class. This approach aimed to enhance the robustness of the machine learning model by providing an expanded and more diverse training dataset.

▹ **Uncertainties in Space Behavior:** Despite valuable insights gained from the experiments on the terrestrial dataset, uncertainties persist regarding the behavior of the dataset in space. The unpredictable nature of the space environment introduces vulnerabilities into the study, emphasizing the necessity for meticulous treatment and analysis during in-orbit tests involving a flight model.

▹ **Flexibility and Adaptability:** In our present investigation, both the training and testing phases of the machine learning model are conducted on Earth, while the runtime inference is exclusively executed by the space system. Nevertheless, the proposed SVM digital training architecture exhibits adaptability and reconfigurability, opening avenues for in-orbit training and modeling. This flexibility proves advantageous when space-collected data significantly differs from terrestrial data, allowing adjustments and re-creation of the SVM model in space. This aspect underscores a fundamental attribute of our work: its flexibility and adaptability, facilitating the execution of training, modeling, and runtime inference in either terrestrial or space environments. However, the associated costs of in-orbit training and modeling, notably in terms of energy consumption and time, should be carefully considered.

▹ **FPGA-Based Digital Implementation:** Concerning the edge-computing component, this study introduces an FPGA-based digital implementation of the SVM machine learning classification model for thermal anomaly detection. While FPGAs offer exceptional resource and capacity capabilities, the discussion highlights their elevated power consumption, necessitating a detailed power budget analysis. In fact, the power budget evaluation is crucial to determine whether the FPGA power demands align with the available power resources from the Electric Power System (EPS), solar panels, and batteries for CubeSats. If the power budget proves inadequate for computing tasks, an alternative hardware solution needs to be explored and tailored to meet mission requirements.

This discussion underscores the complexities and considerations involved in integrating edge computing for thermal anomaly detection in space systems. It emphasizes the importance of adaptability, careful analysis of uncertainties, and careful choice of hardware to ensure the success and reliability of space missions. However, consistency in hardware and software configurations between the flight model and the collection of ground datasets contributes to more predictable performance in space missions.

6. Conclusions

This paper introduces an edge computing approach tailored for space systems, specifically focusing on thermal anomaly detection. The methodology incorporates the SVM algorithm, implemented on an FPGA, which serves as an edge-computing system for onboard data processing. The targeted application involves thermal anomaly detection, employing infrared thermal cameras as an edge-sensing component strategically directed at electronic circuits, generating thermal images of the payload and capturing the associated heat distribution profile. The emphasis lies on leveraging machine learning for thermal anomaly detection in the challenging space environment.

Recognizing the limitations inherent in evaluating the machine learning algorithm on Earth within a controlled environment at ambient temperature, not representative of space conditions, the study employed a strategic approach. A comprehensive evaluation of the SVM classification model was conducted to discern the intrinsic characteristics within each class and highlight their distinctive features. Data augmentation was introduced to introduce variability into the training data, thereby enhancing the robustness and adaptability of the SVM classification model. Although the current study focuses on conducting training and testing on Earth, with inference performed on the space system using a pretrained model, the inclusion of the FPGA-based training architecture provides a comprehensive solution for future scenarios where in-space model reconfiguration or fine-tuning may be necessary. This approach offers flexibility for potential in-orbit training, allowing the model to adjust if significant differences between space-collected and terrestrial data arise, which could be critical for maintaining high performance in space. However, we acknowledge that in-orbit training would entail significant energy and time costs, which must be carefully evaluated for future missions.

In conclusion, this work represents an initial exploration into the integration of edge computing in space systems, coupled with machine learning for thermal anomaly detection. Future endeavors will dedicated to advancing the technology towards space qualification, in-space testing for the development of ready-to-fly software and hardware, and experimentation within a vacuum chamber, replicating the space environment.

Acknowledgments

We express our gratitude to the members of the AI4Space team at the Interdisciplinary Centre for Security, Reliability, and Trust (SnT) at the University of Luxembourg for providing the thermal images that significantly contributed to the research presented in this paper.

References

Ackerman, S., Farchi, E., Raz, O. et al. (2022). Detection of data drift and outliers affecting machine learning model performance over time. URL: https://arxiv.org/abs/2012.09258. arXiv:2012.09258.

Ackerman, S., Raz, O., Zalmanovici, M. et al. (2021). Automatically detecting data drift in machine learning classifiers. URL: https://arxiv.org/abs/2111.05672. arXiv:2111.05672.

Al-Rababah, K., Mustaffa, M. R., Doraisamy, S. C. et al. (2021). Hybrid discrete wavelet transform and histogram of oriented gradients for feature extraction and classification of breast dynamic thermogram sequences. In *2021 Fifth International Conference on Information Retrieval and Knowledge Management (CAMP)* (pp. 31–35). doi:10.1109/CAMP51653.2021.9498028.

Babuscia, A., Corbin, B., Knapp, M. et al. (2013). Inflatable antenna for cubesats: Motivation for development and antenna design. *Acta Astronautica*, *91*, 322–332.

Bottou, L. (2010). Large-scale machine learning with stochastic gradient descent. In Y. Lechevallier, & G. Saporta (Eds.), *Proceedings of COMPSTAT'2010* (pp. 177–186). Heidelberg: Physica-Verlag HD.

Bottou, L., & Lin, C.-J. (2007). Support vector machine solvers. (pp. 301–320). doi:10.7551/mitpress/7496.003.0003.

Bouwmeester, J., & Guo, J. (2010). Survey of worldwide pico-and nanosatellite missions, distributions and subsystem technology. *Acta Astronautica*, *67*(7-8), 854–862.

Bratvold, T. (2022). Near real-time hyperspectral image classification for in-orbit decissionmaking hypso-1. *NTNU*, .

Buttazzoni, G., Comisso, M., Cuttin, A. et al. (2017). Reconfigurable phased antenna array for extending cubesat operations to ka-band: Design and feasibility. *Acta Astronautica*, *137*, 114–121.

Cervantes, J., Li, X., Yu, W. et al. (2008). Support vector machine classification for large data sets via minimum enclosing ball clustering. *Neurocomputing*, *71*(4), 611–619. URL: https://www.sciencedirect.com/science/article/pii/S0925231207002962. doi:https://doi.org/10.1016/j.neucom.2007.07.028. Neural Networks: Algorithms and Applications 50 Years of Artificial Intelligence: a Neuronal Approach.

Chang, C.-C., & Lin, C.-J. (2007). Libsvm: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, *2*.

Chen, J., Pi, D., Wu, Z. et al. (2021). Imbalanced satellite telemetry data anomaly detection model based on bayesian lstm. *Acta Astronautica*, *180*, 232–242.

Colin, A., Ruppel, E., & Lucia, B. (2018). A reconfigurable energy storage architecture for energy-harvesting devices. ASPLOS '18 (p. 767781). New York, NY, USA: Association for Computing Machinery. URL: https://doi.org/10.1145/3173162.3173210. doi:10.1145/3173162.3173210.

Dalal, N., & Triggs, B. (2005). Histograms of oriented gradients for human detection. In *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)* (pp. 886–893 vol. 1). volume 1. doi:10.1109/CVPR.2005.177.

Denby, B., & Lucia, B. (2020). Orbital edge computing: Nanosatellite constellations as a new class of computer system. In *Proceedings of the Twenty-Fifth International Conference on Architectural Support for Programming Languages and Operating Systems* (p. 939954). New York, NY, USA: Association for Computing Machinery. URL: https://doi.org/10.1145/3373376.3378473.

Ergen, T., & Kozat, S. S. (2020). A novel distributed anomaly detection algorithm based on support vector machines. *Digital Signal Processing*, *99*, 102657. URL: https://www.sciencedirect.com/science/article/pii/S1051200420300026. doi:https://doi.org/10.1016/j.dsp.2020.102657.

Giuffrida, G., Fanucci, L., Meoni, G. et al. (2022). The -sat-1 mission: The first on-board deep neural network demonstrator for satellite earth observation. *IEEE Transactions on Geoscience and Remote Sensing*, *60*, 1–14. doi:10.1109/TGRS.2021.3125567.

Heller, K. A., Svore, K. M., Keromytis, A. D. et al. (2003). One class support vector machines for detecting anomalous windows registry accesses.

Ibrahim, S. K., Ahmed, A., Zeidan, M. A. E. et al. (2018). Machine learning methods for spacecraft telemetry mining. *IEEE Transactions on Aerospace and Electronic Systems*, *55*(4), 1816–1827.

Jayaraman, V., Chandrasekhar, M., & Rao, U. (1997). Managing the natural disasters from space technology inputs. *Acta Astronautica*, *40*(2), 291–325. URL: https://www.sciencedirect.com/science/article/pii/S009457659700101X. doi:https://doi.org/10.1016/S0094-5765(97)00101-X. Enlarging The Scope of Space Applications.

Kacker, S., Meredith, A., Cahoy, K. et al. (2022). Machine learning image processing algorithms onboard ops-sat.

Keerthi, S. S., & Lin, C.-J. (2003). Asymptotic Behaviors of Support Vector Machines with Gaussian Kernel. *Neural Computation*, *15*(7), 1667–1689. URL: https://doi.org/10.1162/089976603321891855. doi:10.1162/089976603321891855.

Labrèche, G., Evans, D., Marszk, D. et al. (2022). Ops-sat spacecraft autonomy with tensorflow lite, unsupervised learning, and online machine learning. In *2022 IEEE Aerospace Conference (AERO)* (pp. 1–17). doi:10.1109/AERO53065.2022.9843402.

Laib dit Leksir, Y., Mansour, M., & Moussaoui, A. (2018). Localization of thermal anomalies in electrical equipment using infrared thermography and support vector machine. *Infrared Physics & Technology*, *89*, 120–128. URL: https://www.sciencedirect.com/science/article/pii/S1350449517306631. doi:https://doi.org/10.1016/j.infrared.2017.12.015.

Lakshminarayanan, V., & Sriraam, N. (2014). The effect of temperature on the reliability of electronic components. (pp. 1–6). doi:10.1109/CONECCT.2014.6740182.

scikit learn (). Support Vector Machines.

Li, Q., Wang, S., Ma, X. et al. (2021). Service coverage for satellite edge

computing. *IEEE Internet of Things Journal*, *9*(1), 695–705.

Li, Y., & Su, G. (2015). Simplified histograms of oriented gradient features extraction algorithm for the hardware implementation. In *2015 International Conference on Computers, Communications, and Systems (ICCCS)* (pp. 192–195). doi:10.1109/CCOMS.2015.7562899.

Manning, C. D., Raghavan, P., & Schütze, H. (2008). *Introduction to Information Retrieval*. Cambridge, UK: Cambridge University Press.

Mateo-Garcia, G., Veitch-Michaelis, J., Smith, L. et al. (2021). Towards global flood mapping onboard low cost satellites with machine learning. *Scientific Reports*, *11*. doi:10.1038/s41598-021-86650-z.

Meoni, G., Märtens, M., Derksen, D. et al. (2024). The ops-sat case: A data-centric competition for onboard satellite image classification. *Astrodynamics*, (pp. 1–22). doi:10.1007/s42064-023-0196-y.

Nalepa, J., Myller, M., Andrzejewski, J. et al. (2022). Evaluating algorithms for anomaly detection in satellite telemetry data. *Acta Astronautica*, *198*, 689–701.

Santoni, F., Piergentili, F., Donati, S. et al. (2014). An innovative deployable solar panel system for cubesats. *Acta Astronautica*, *95*, 210–217.

Selva, D., & Krejci, D. (2012). A survey and assessment of the capabilities of cubesats for earth observation. *Acta Astronautica*, *74*, 50–68.

Shakarami, A., Ghobaei-Arani, M., & Shahidinejad, A. (2020). A survey on the computation offloading approaches in mobile edge computing: A machine learning-based perspective. *Computer Networks*, *182*, 107496.

Systems, F. (2018). LWIR Micro Thermal Camera Module Lepton Datasheet. https://www.mouser.es/datasheet/2/227/Lepton3_3_5_Data_Sheet-2580086.pdf.

Tsang, I. W., Kwok, J. T., & Cheung, P.-M. (2005). Core vector machines: Fast svm training on very large data sets. *Journal of Machine Learning Research*, *6*(13), 363–392. URL: http://jmlr.org/papers/v6/tsang05a.html.

Vapnik, V. N. (2000). The nature of statistical learning theory. In *Statistics for Engineering and Information Science*.

Yang, J., Wang, W., Lin, G. et al. (2019). Infrared thermal imaging-based crack detection using deep learning. *IEEE Access*, *7*, 182060–182077. doi:10.1109/ACCESS.2019.2958264.

Zhang, S., Wang, P., Wan, Y. et al. (2022). V/ka-band leo high-throughput satellite and integrated satellite–terrestrial network experiment system: First two years flight results. *Acta Astronautica*, *201*, 533–553.

Zhu, X. (2010). Support vector machines. In *CS769 Spring 2010 Advanced Natural Language Processing*.

Declaration of interest statement:

Me, Carmen MISA MOREIRA, declare that there is no financial and/or personal relationships with other organisations that could influence my work.