FPGA-Enabled Machine Learning Applications in Earth Observation: A Systematic Review

CÉDRIC LÉONARD, Technical University of Munich, Germany and Remote Sensing Technology Institute (IMF), German Aerospace Center (DLR), Germany
DIRK STOBER, Technical University of Munich, Germany
MARTIN SCHULZ, Technical University of Munich, Germany

New UAV technologies and the NewSpace era are transforming Earth Observation missions and data acquisition. Numerous small platforms generate large data volume, straining bandwidth and requiring onboard decision-making to transmit high-quality information in time. While Machine Learning allows real-time autonomous processing, FPGAs balance performance with adaptability to mission-specific requirements, enabling onboard deployment. This review systematically analyzes 66 experiments deploying ML models on FPGAs for Remote Sensing applications. We introduce two distinct taxonomies to capture both efficient model architectures and FPGA implementation strategies. For transparency and reproducibility, we follow PRISMA 2020 guidelines and share all data and code at https://github.com/CedricLeon/Survey_RS-ML-FPGA.

CCS Concepts: • **General and reference** → **Surveys and overviews**; • **Computing methodologies** → *Machine learning*; • **Hardware** → *Reconfigurable logic and FPGAs*; • **Applied computing** → *Earth and atmospheric sciences.*

Additional Key Words and Phrases: Earth Observation, Remote Sensing, Neural Networks, Approximate Computing

ACM Reference Format:
Cédric Léonard, Dirk Stober, and Martin Schulz. 2025. FPGA-Enabled Machine Learning Applications in Earth Observation: A Systematic Review. In *Proceedings of Make sure to enter the correct conference title from your rights confirmation emai (Conference acronym 'XX).* ACM, New York, NY, USA, 35 pages. https://doi.org/XXXXXXX.XXXXXXX

Authors' Contact Information: Cédric Léonard, cedric.leonard@dlr.de, Technical University of Munich, Munich, Germany and Remote Sensing Technology Institute (IMF), German Aerospace Center (DLR), Weßling, Germany; Dirk Stober, dirk.stober@tum.de, Technical University of Munich, Munich, Germany; Martin Schulz, schulzm@in.tum.de, Technical University of Munich, Munich, Germany.

1 Introduction

The last decade of research has seen major innovations in Machine Learning (ML), particularly in Deep Learning (DL). Every discipline has seen publications leveraging ML's abstract learning capabilities to alleviate specific problems of the field. Nowadays, many state-of-the-art solutions incorporate ML methods in their pipeline. For instance, automatic navigation, for cars and Unmanned Aerial Vehicles (UAVs) alike, heavily relies on scene segmentation and obstacle detection. Similarly, video and image encoding apply learnable heuristics to select optimal patterns and splits in the data. Beyond vision, language processing benefits from the wide context available through transformer tokenization, significantly enhancing performance. By and large, these AI technologies have significant computational costs, making them inadequate in resource-constrained environments. Therefore, many applications deploying their algorithms close to the sensor, e.g., cars and UAVs On-Board Computers (OBCs), GoPros, or intelligent microphones, benefit from lightweight optimizations of these computationally intensive models. Approximate computing optimizations, trading computation load over accuracy, such as quantization, can typically reduce memory overhead by a factor of 4 and computational costs by a factor of 16 [85].

In this survey, we take a systematic look at one application of ML for computer vision: Remote Sensing (RS) for Earth Observation (EO). Contemporary challenges, such as the climate crisis, urban planning, or defense concerns, transformed RS into an essential field. In particular, the rising NewSpace[1] industry has decreased the costs and the production time of SmallSats. Because deploying CubeSats has never been so easy, we can observe a steady increase in the number of missions—to date, over 2600 nanosatellites have been launched [63]. However, payload restrictions on these compact satellites reduce their onboard compute capacity relative to their larger counterparts, forcing them to run application-specific routines in flight and underscoring the need for efficient onboard data processing.

In the particular example of RS, while the quantity of data grows, the bandwidth resources for the data transmission (downlink) stagnate [95]. Because of such limitations, most current missions acquire data on-demand or over certain pre-configured conditions, inevitably missing out relevant information. The increasing sensing resolution and number of in-orbit satellites, paired with further restrictions on the radio-frequency spectrum, will worsen the problem [103]. Processing the data directly onboard, transmitting only compressed or application-tailored information, constitutes an elegant solution for future missions. Equipping future missions with edge computing platforms opens up possibilities to relieve data processing stress. Because SmallSats' standard payloads typically include FPGA-based hardware [33, 110], this survey focuses on analyzing the literature deploying ML-based solutions for RS applications on FPGAs.
The main contributions of this survey are:

- **Systematic literature collection and analysis** ensuring a rigorous and transparent review process following the PRISMA 2020 guidelines.
- **Two specialized taxonomies**, one structuring RS applications based on their ML problem formulation, and another categorizing FPGA implementations by frameworks and design patterns.
- **Comprehensive evaluation** of existing works, structured around eight Research Questions (RQs) to analyze the research landscape and method synergies.
- **Identification of research gaps**, structured under five subtopics to guide future advancements in FPGA-enabled ML for Earth Observation.
- **Guidelines and recommendations** to facilitate benchmarking, enhance reproducibility, and drive future research efforts.

[1]https://philab.esa.int/tag/new-space/

1.1 Background

Remote Sensing (RS)[2] is the science of obtaining reliable information about objects or areas on the Earth's surface without direct physical contact [14]. It involves the acquisition, processing, and interpretation of data captured by sensors that detect electromagnetic radiation. The discipline is broadly divided into active and passive approaches. Active systems, such as radar and LiDAR, emit their own energy signals and record the returning echoes, while passive systems depend on external sources of energy, primarily sunlight. Furthermore, RS platforms are classified by their operational altitude into spaceborne and airborne systems. Spaceborne platforms, typically mounted on satellites, provide extensive temporal and spatial coverage. They enable continuous monitoring over vast areas and across diverse time scales—from near real-time updates to the analysis of long-term environmental trends [145]. Airborne platforms, on the other hand, offer higher spatial resolution over more localized areas, making them ideal for detailed studies that require fine-grained spatial information.

Machine Learning (ML) methods have existed for several decades, but the past ten years have seen an unprecedented surge in their capabilities, largely driven by advances in computing power. On one hand, Deep Learning (DL) abilities to explore and learn from high-dimensional spaces have precipitated the growth of the discipline. In particular, Foundation Models and their generalization powers have been an intense research area over the last two years [115]. On the other hand, traditional ML methods such as Decision Trees (DT) [96] or Support Vector Machines (SVM) [25] are small and explainable methods still preferred in certain scenarios. Nevertheless, the vast majority of current ML research focuses on DL, equally in RS, where Convolutional Neural Network (CNN) and Vision Transformer (ViT) abilities to learn from images lay a solid basis for many RS applications [78, 115].

CPUs and, to a lesser extent, GPUs support a wide range of algorithms and kernels, requiring general-purpose hardware units such as multi-level caches, prefetching, and branch prediction. Many specific algorithms do not fully utilize these features, leading to wasted energy and transistors. FPGAs, by contrast, are reconfigurable integrated circuits adaptable to a specific algorithm, instantiating only the necessary hardware units. FPGAs' reconfigurability comes at the cost of lower operating frequency, requiring comparatively higher parallelism to compete with CPUs and the already highly parallel GPUs. In addition, such a high degree of freedom requires careful development. Programming FPGAs is similar to designing an integrated circuit and is traditionally done at low-level using Hardware Description Languages (HDL). However, with the growing adoption of AI on FPGAs, higher-level end-to-end toolchains are now available to implement Neural Networks (NNs) with popular kernels, such as CNNs [5, 118].

1.2 Related Work

We conduct this survey to combine the overlapping disciplines of Remote Sensing, Machine Learning, and FPGA deployment. To the authors' knowledge, no work has yet explored the intersection of these three research domains. However, multiple reviews have covered the usage of ML in RS, explored FPGAs for onboard processing, or investigated FPGA-based accelerators for ML separately—see Table 1 for some relevant literature reviews. Notably, two surveys investigate the overlap between DL and FPGA design: Abdelouahab et al. [2] and Shawahna et al. [108]. Both studies draft a comprehensive overview of design paradigms to implement DL methods on FPGAs. Taking a closer look at onboard environments, George and Wilson [33] and Bouhali et al. [12] depict the specificities of onboard flying platforms, space- and airborne, respectively. With a similar scope, Lange et al. [67] specifically focus on the impact and mitigation techniques of radiation effects on DL models.

[2]Throughout this survey we will prefer the term Remote Sensing (RS) rather than Earth Observation (EO). RS specifically refers to acquiring information about the Earth's surface from a distance while EO includes both remote sensing and in situ measurements.

Table 1. Summary of related surveys. ✓ covered, × not covered, ≈ partially covered.

Reference	Year	RS Focus	ML Focus	FPGAs	Onboard	Years analyzed
Lopez et al. [79]	2013	✓	×	✓	Space	–
Bouhali et al. [12]	2017	≈	×	✓	UAV	–
Zhu et al. [145]	2017	✓	DL	×	×	2014–2017
Abdelouahab et al. [2]	2018	×	CNNs	✓	×	2015–2018
George and Wilson [33]	2018	≈	×	✓	Space	2000–2016
Shawahna et al. [108]	2019	×	DL	✓	×	2009–2018
Osco et al. [92]	2021	✓	DL	×	UAV	2012–2020
Lange et al. [67]	2024	≈	DL	×	Space	2018–2023
Our survey	2025	✓	DL + ML	✓	✓	2014–2024

2 Methodology

This study follows the Preferred Reporting Items for Systematic Reviews and Meta-Analyses (PRISMA) 2020 guidelines [93] applicable to Computer Science, ensuring a complete, transparent, and reproducible review process.

2.1 Research Questions

This survey aims to examine articles using ML methods on FPGA hardware and discussing RS in EO. We specifically focus on RS sensors mounted on aerial vehicles (drones or UAVs) or spaceborne satellites. We explore this research intersection to summarize the community's work towards implementing automated methods onboard flying platforms, space- or airborne. To provide a clear structure for this survey, we address the following eight Research Questions (RQs). First, in Section 3, RQ1-3 establish the research landscape and define the population of studies included in our analysis.

- RQ1: Which Remote Sensing applications are addressed, and how are they formulated as ML problems?
- RQ2: Which Machine Learning models are most prevalent, and what are their key architectural characteristics?
- RQ3: What are the motivations for choosing FPGAs over other computing platforms (CPUs, GPUs, ASICs)?

Then in Section 4, RQ4-6 explore the technical details of the deployed solutions. Here, we catalog ML optimization methods, FPGA design paradigms, and various levels of parallelism available to designers.

- RQ4: What are the effective optimization strategies to reduce model footprint and computational complexity?
- RQ5: How do FPGA design frameworks and patterns affect development effort and final performance?
- RQ6: How do different FPGA design approaches, such as pipelining, parallelization, and memory optimization, balance performance and constraints?

Finally in Section 5, R7-8 take a holistic view of the accomplishments in this field, discussing key achievements and, in line with the current NewSpace era, the potential of AI on the edge for EO missions.

- RQ7: What combination of Machine Learning model, optimization strategies, and FPGA design best synergizes to optimize key Remote Sensing metrics?
- RQ8: How are AI-powered edge computing solutions transforming EO missions?

To answer these research questions, we conduct a systematic search procedure adapted from the method proposed by the PRISMA 2020 guidelines [93]. The procedure starts by defining a prompt to frame for the population of records of interest. After collecting the results from the chosen research databases, we perform an initial screening pass via metadata analysis to filter out undesired records. The second screening pass involves reviewing the complete records, excluding any not clearly identified as being outside the survey's scope based on the metadata alone.

2.2 Search Procedure

We systematically retrieve relevant studies by querying Web of Science[3] with the prompt[4] below:

TOPICS: [*(Machine Learning* ***OR*** *Deep Learning* ***OR*** *Neural Network* ***OR*** *Convolutional Neural Network* ***OR*** *Recurrent Neural Network* ***OR*** *ML* ***OR*** *DL* ***OR*** *NN* ***OR*** *CNN* ***OR*** *RNN)*

AND

(Earth Observation ***OR*** *Remote Sensing* ***OR*** *Earth Science* ***OR*** *LiDAR* ***OR*** *SAR* ***OR*** *UAV* ***OR*** *Sentinel)*

AND

(FPGA ***OR*** *Field-Programmable Gate Array* ***OR*** *Field Programmable Gate Array* ***OR*** *Versal)*]

YEAR PUBLISHED: 2014-2024

LANGUAGE: *ENGLISH*

The central part of the prompt consists of three expressions framing for any ML model, the usage of RS data in EO, and the explicit mention of FPGA devices. We choose not to consider articles published more than 10 years ago. Indeed, the FPGA technology, the state-of-the-art DL models, and even the tackled problems have evolved rapidly. Therefore, work from more than 10 years ago is often too different from modern studies to be compared fairly.

After retrieving the 119 results of the prompt, we proceed to screen the articles to filter mismatches. To this effect, we pick 4 Exclusion Criteria (EC):

- EC1 - Discussion and review papers, without experiments.
- EC2 - Publication unrelated to or not using ML models;
- EC3 - Publication unrelated to or not using RS data for EO;
- EC4 - Publication unrelated to or not using FPGA devices;

While EC1 and EC2 are self-explanatory, EC3 mostly rules out non-EO applications of remote sensing technology, e.g., LiDAR for autonomous driving or SAR calibration. EC4 excludes studies with no experiments or details about the FPGA implementation. For example, EC4 eliminates records mentioning FPGA hardware as a possible solution or using a different FPGA acronym, e.g., fast patch-free global learning. We design this filtering process to retain only the most relevant, experiment-based studies that reflect the current state of FPGA-based ML in Remote Sensing.

After screening, 46 articles are kept for review. Each article included in the survey is thoroughly read and tagged in Zotero [24]. In particular, we use Zotero's tagging system and API to systematically report the experiments' specificities and performance metrics. The entire database extracted via Zotero's API, along with the code used for analysis, figures, and main taxonomy tables, is available on GitHub at https://github.com/CedricLeon/Survey_RS-ML-FPGA. Our repository also hosts supplemental material, such as extended methodological considerations.

[3]https://www.webofscience.com/, last consulted on 06 March 2025

[4]The "**TOPICS**" field searches across each item's title, abstract, author keywords, and keyword plus in the database. The prompt is case-insensitive.

3 Mapping the Research Landscape: Applications, Models, and Hardware

In this section, we analyze the motivations, tackled problems, and solutions of each study included in the survey. In particular, we propose a taxonomy based on the ML and RS components of the experiments in Table 2.

3.1 Remote Sensing/Machine Learning Taxonomy

To help readers explore studies addressing similar problem settings, we group the collected studies by how RS applications are formulated as ML tasks. Table 2 presents this taxonomy, clustering all reported experiments across the selected studies. When a study conducts several relevant experiments, we extract each corresponding model and metric. For example, Suh et al. [113] present a methodological contribution and evaluate several sizes of the same architecture, i.e., SSD 0.25x, SSD 0.5x, and SSD 1.0x. We argue that reporting each experiment separately, rather than considering a single experiment per study, portrays a more complete picture of the conducted work. In total, the taxonomy includes 66 experiments drawn from 46 studies.

Each row in Table 2 corresponds to an experiment. The first group of columns, dedicated to the *RS Problem*, introduces the taxonomy: experiments are hierarchically grouped by ML *'Task'*, i.e., the family of the ML problem, data modality (*'Mod.'*), i.e., the type of RS data used as input, *'Application'* and *'Dataset'*. As a result, experiments from the same study may not be juxtaposed. *'Application'* refers to the downstream task tackled by the study; applications are closely tied with the *'Dataset'*[5] and are detailed later in Fig. 1. The *'Article'* group provides context: *'Ref.'* lists the source article and *'Year'* indicates its publication date.

The rest of the table is dedicated to the ML component of the experiment. *'ML Model'* characterizes the model's architecture through three fields: *'Original Name'*, the exact appellation used in the study; *Core*, a high-level architectural descriptor (e.g., CNN, Shallow NN such as a Multi-Layer Perceptron (MLP), or Traditional ML like an SVM), further detailed in Fig. 2; and, for Deep Neural Networks (DNNs), their *'Backbone'*. Next, the *'FPGA'* column indicates the family of the FPGA device used for implementation, more details in Fig. 3. The last group of columns *'Performance'* reports a few metrics of the experiment: *'Score'* corresponds to the accuracy in **[%]** of the model on its dataset and *'MF**[MB]**'* is the Memory Footprint of the model in MegaBytes. Finally, *'C**[OP]**'* stands for the computational complexity of the model in numbers of operations[6]. These performance metrics are the most closely related to the ML model; further metrics are reported during the second literature taxonomy in Table 3.

3.2 Formulating RS applications as ML problems (RQ1)

From urban planning to disaster assessment and target monitoring, Remote Sensing images support a wide and growing range of Earth Observation applications [142].

Overview of RS Applications. Fig. 1 separates the surveyed RS applications into four distinct thematics. *Target Surveillance*[7] gathers all applications focused on recognizing a specific family of objects. The output of such a task can be the location—bounding box—of the targets, in such a case we denote it detection (det.), or the binary presence of the target in the observed image, denoted identification (id.). In this thematic, "Miscellaneous targets" refers to heterogeneous object families, such as ships and cars, but also tennis courts or roundabouts in the DOTAv1.0 [128] or DIOR [74] datasets. *Landscape analysis* applications investigate the type of terrain present in the image. In this survey,

[5]Datasets carrying the *(cust.)* tag are unpublished datasets, meaning that the authors built their dataset but did not name or publish it.
[6]More details about the distinction between **[GOP/s]**, **[GOP]**, or **[GOPs]** can be found in Section 6.3: Recommendations.
[7]*Target detection* is a more frequent appellation for RS applications. However, we will prefer the term "surveillance" to avoid any confusion with the ML task "object detection," sometimes also called "target detection."

Table 2. RS/ML Taxonomy Table

RS Problem				Article		ML Model			FPGA	Performance		
Task	**Mod.**	**Application**	**Dataset**	**Ref.**	**Year**	**Original Name**	**Core**	**Backbone**	**Family**	**Score [%]**	**MF[MB]**	**C[OP]**
Classification	RGB	Landcover/Land use	UCM Land Use [134]	[139]	2023	A2NN	VGG	VGG11	Virtex-7	94.76 (OA)	-	14.99G
			NWPU-RESISC45 [20]	[131]	2022	Improved VGG16	VGG	VGG16	Artix-7	88.08 (OA)	14.80	40.96G
				[141]	2020	Q-IORN	VGG	VGG16	Virtex-7	88.31 (OA)	121.51	-
				[89]	2023	ResNet-34	ResNet	ResNet-34	Virtex-7	92.81 (OA)	21.29	7.33G
				[89]	2023	VGG16	VGG	VGG16	Virtex-7	91.90 (OA)	14.70	30.69G
			UAV RGB (cust.)	[82]	2019	MLP	Shallow NN	-	Spartan-3A	95.14 (OA)	-	-
		Ship id.	MASATI [30]	[86]	2022	AlexNetLite	AlexNet	AlexNet	Kintex US	89 (F1)	-	5.71G
				[86]	2022	MobileNetv1Lite	Custom CNN	MobileNetv1	Kintex US	94 (F1)	-	0.05G
			Kaggle SSI	[55]	2024	HO-ShipNet	Custom CNN	-	Zynq 7000	93.44 (F1)	-	-
		Cloud coverage	L8 Biome [27]	[95]	2022	CloudSatNet-1 Q2	Custom CNN	-	Zynq 7000	83.41 (OA)	1.43	-
				[95]	2022	CloudSatNet-1 Q4	Custom CNN	-	Zynq 7000	87.42 (OA)	3.06	-
			RGB (cust.)	[119]	2024	ResNet-50	ResNet	ResNet-50	Zynq US+	-	-	-
				[119]	2024	SICNet	Custom CNN	-	Zynq US+	85.76 (OA)	-	-
			Landsat-8 (cust.)	[63]	2024	TriCloudNet + U-Net	U-Net	SqueezeNet	Zynq 7000	6.84 (CRE)	3.61	-
				[121]	2024	CNN	Custom CNN	-	Artix-7	89.64 (F1)	-	-
	HSI	Air quality mon.	*UAV RGB (cust.)*	[53]	2024	ETAUS	ResNet	ResNet-50	Zynq US+	86.38 (F1)	-	-
		Cloud coverage	*Sentinel-2 (cust.)*	[98]	2021	CloudScout	Custom CNN	-	Zynq US+	92.0 (OA)	13.30	-
	SAR	Military targets id.	MSTAR [107]	[126]	2019	LeNet-5 f32	LeNet	LeNet-5	Kintex-7	98.76 (OA)	6.64	-
				[126]	2019	LeNet-5 i8	LeNet	LeNet-5	Kintex-7	97.77 (OA)	1.66	-
				[18]	2020	LeNet-5	LeNet	LeNet-5	Kintex-7	98.18 (OA)	-	-
				[137]	2022	GNN	GNN	-	Zynq US+	99.09 (OA)	0.96	2.13M
				[138]	2023	GNN	GNN	-	Alveo	-	-	-
		Ship id.	*ALOS-2 (cust.)*	[46]	2019	CNN	Custom CNN	-	Zynq 7000	100 (OA)	-	-
				[84]	2020	BNN	Custom CNN	-	Zynq 7000	-	-	-
Pixel classification	RGB	Deforestation det.	*UAV RGB (cust.)*	[117]	2020	Weightless Neural Systems	Shallow NN	-	Zynq 7000	90.0 (OA)	-	-
		UAV landing site id.	*UAV RGB (cust.)*	[28]	2018	Decision Tree	Trad. ML	-	Zynq 7000	92.1 (OA)	-	-
		Landcover/Land use	*ALSAT-2A (cust.)*	[130]	2017	BRAM_DSP	Trad. ML	-	Virtex-6	94.8 (OA)	-	-
				[130]	2017	LUT_MUL	Trad. ML	-	Virtex-6	94.8 (OA)	-	-
	HSI	Landcover/Land use	University of Pavia [120]	[80]	2024	SVM	Trad. ML	-	Zynq 7000	82.48 (AA)	-	-
				[48]	2023	2D CNN	Custom CNN	-	Zynq US+	98.24 (OA)	1.20	0.684M
				[48]	2023	3D CNN	Custom CNN	-	Zynq US+	94.09 (OA)	0.12	4.231M
				[48]	2023	HybridSN	Custom CNN	-	Zynq US+	100 (OA)	20.50	101.642M
				[109]	2021	LPDBL	Trad. ML	-	Virtex-7	-	-	-
				[16]	2024	SAM-GNN	GNN	-	Virtex-7	95.05 (OA)	-	-
			AVIRIS-NG	[42]	2022	CAG-SC2S	Trad. ML	-	Artix-7	98.30 (OA)	-	-
				[42]	2022	CAL-SC2S	Trad. ML	-	Artix-7	99.69 (OA)	-	-
				[42]	2022	SVM	Trad. ML	-	Artix-7	20.71 (OA)	-	-
		Anomaly det.	ABU [62]	[13]	2023	Deep Belief Network	Shallow NN	-	Zynq US+	-	-	-
Segmentation	RGB	Urban areas	Potsdam [57]	[104]	2021	ENet	Custom CNN	-	Zynq US+	63.3 (mIoU)	0.36	4.06G
				[104]	2021	ESPNet	Custom CNN	-	Zynq US+	55.5 (mIoU)	0.33	3.71G
				[104]	2021	FPN	Custom CNN	-	Zynq US+	65.3 (mIoU)	5.84	17.30G
				[104]	2021	U-Net	U-Net	-	Zynq US+	61.4 (mIoU)	7.40	96.68G
		General	*Landsat-8-OLI (cust.)*	[99]	2021	RDBC	Trad. ML	-	Virtex-6	37.45 (SSIM)	-	O(n)
		Cloud coverage	38-Cloud [83]	[8]	2019	C-FCN++	Custom CNN	-	Cyclone V	79.30 (mIoU)	0.047	4.7K
Object detection	RGB	Diverse det.	DOTAv1.0 [128]	[140]	2021	Improved YOLOv2	YOLO	Darknet19	Zynq 7000	67.32 (mAP)	-	379G
				[131]	2022	Improved YOLOv2	YOLO	Darknet19	Artix-7	67.30 (mAP)	49.40	379.55G
				[89]	2023	Improved YOLOv2	YOLO	Darknet19	Virtex-7	67.30 (mAP)	49.40	379.55G
				[133]	2023	Ghost-YOLOS	YOLO	GhostNet	Zynq 7000	62.58 (mAP)	12.60	8.41G
			DIOR [74]	[143]	2023	YOLOv4-MobileNetv3	YOLO	MobileNetv3	Zynq US+	82.61 (mAP)	5.69	-
				[71]	2023	RFA-YOLO	YOLO	MobileNeXt	Zynq US+	64.85 (mAP)	24.75	-
			NWPU VHR-10 [21]	[75]	2019	CBFF-SSD	SSD	MobileNetv1	Zynq 7000	91.42 (mAP)	-	-
			DAC 2018	[124]	2019	CNN	Custom CNN	-	Zynq 7000	21 (mIoU)	-	-
			UAV RGB (cust.)	[76]	2020	AP2D-Net	Custom CNN	-	Zynq US+	55.0 (mIoU)	-	-
		Flying-object det.	PennSyn2Real [88]	[113]	2021	SSD 0.25x	SSD	VGG16	Zynq US+	76.2 (mAP)	-	4.04G
				[113]	2021	SSD 0.5x	SSD	VGG16	Zynq US+	83.9 (mAP)	-	15.65G
				[113]	2021	SSD 1.0x	SSD	VGG16	Zynq US+	88.4 (mAP)	-	61.60G
			UAV RGB (cust.)	[87]	2024	YOLOv4	YOLO	Darknet53	Zynq US+	83.7 (mAP)	245	105.9G
				[87]	2024	YOLOv4-tiny 3L	YOLO	Darknet53-tiny	Zynq US+	72.24 (mAP)	24.10	15.32G
		Aircraft det.	Airbus	[31]	2024	ResNet-18+YOLOv2	YOLO	ResNet-18	Zynq US+	91 (AP)	-	-
			GES RGB (sim.)	[127]	2021	Improved YOLOv3	YOLO	DarkNet53	Zynq US+	92 (OA)	8.50	-
		UAV obstacles det.	*UAV RGB+MMW (cust.)*	[125]	2024	Lightweight CNN	ResNet	ResNet-18	Zynq 7000	70.1 (mAP)	-	4.6G
		Railway defect det.	*FastenerDataset (cust.)*	[136]	2024	Improved YOLOv4-tiny	YOLO	Darknet53-tiny	Zynq US+	95.1 (mAP)	-	-
	SAR	Ship det.	SSDD [73]	[132]	2022	CNN2@0.7	YOLO	MobileNetv1	Virtex-7	94 (AP)	0.09	0.45G
				[132]	2022	CNN4@0.7	YOLO	MobileNetv2	Virtex-7	93.3 (AP)	0.11	0.56G
				[132]	2022	CNN6@1.6	YOLO	SqueezeNet	Virtex-7	92.8 (AP)	0.09	0.6G
Regr.	SAR	Oil spills mon.	*SAR (sim.)*	[43]	2022	MLP	Shallow NN	-	Zynq 7000	-	-	-

Task: **Regr.** Regression; *Dataset:* **GES** Google Earth Studio, **MMW** MilliMeter-Wave radar; *Model:* **RDBC** Roller Dung Bettle Clustering, **WNS** Weightless Neural System

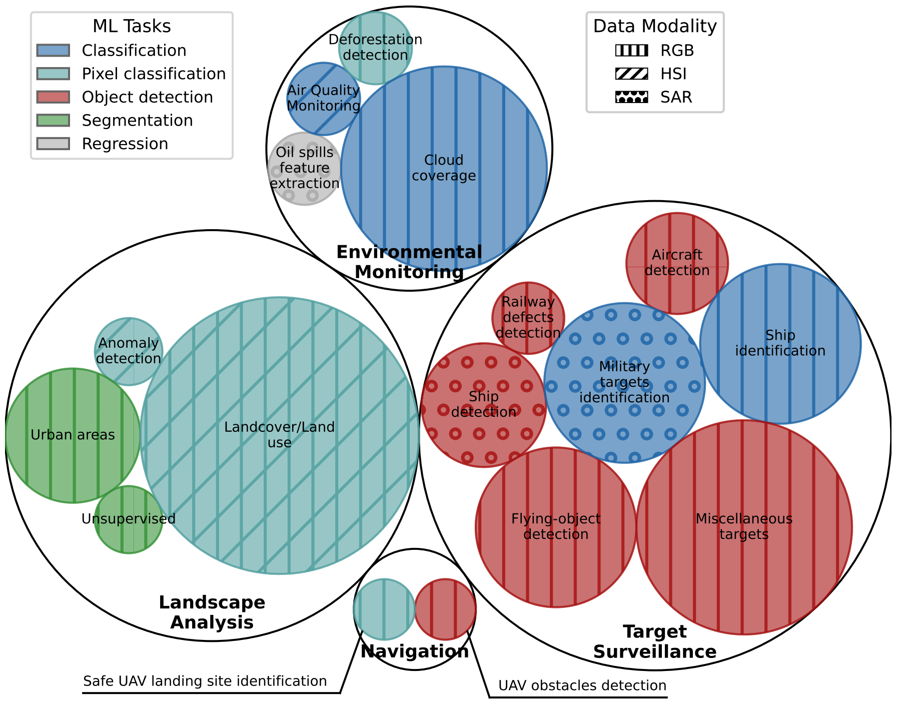

Fig. 1. Addressed RS applications grouped by thematic and colored as their corresponding ML task. The size of the circular patches represents the number of experiments, while the hatching pattern represents the prevailing data modality.

"Urban areas" refers to segmenting city tiles, e.g., with the POTSDAM dataset [57]. "Landcover/Land use" is synonym to the classification of the type of terrain, such as desert, stadium, sea ice, etc., for example, in NWPU-RESISC45 [20]. Finally, *Environment monitoring* includes landscape analysis applications specifically focused on environmental phenomena, such as cloud or deforestation detection. While the grouping of Fig 1 offers a clear structure for the surveyed studies, it does not capture the full range of RS applications. In particular, the records miss prevalent applications such as natural disaster monitoring or studies of the climate crisis.

Data Modalities in RS. The hash pattern in Fig. 1 represents the modality of the images used for each application. In RS, the modality refers to the type of sensor that acquires the data. The most common modality, as in this survey, is optical data, with the three traditional Red-Green-Blue (RGB) channels. In RS, channels are commonly referred to as bands. Indeed, sensors capture data around specific wavelengths, each wavelength's spectrum resulting in a different band. Hyperspectral Imaging (HSI) emerges as the most extreme case, sometimes with over 200 bands, providing a very fine-grained spectral resolution that approximates a contiguous spectrum [79]. Lastly, Synthetic Aperture Radar (SAR) is an active sensor, in opposition to passive sensors that collect light from the observed scene. SAR emits waves that scatter upon hitting the observed scene, then captures the reflected components that bounce back towards the sensor. Such a system allows to collect data during the night and through clouds, making SAR a versatile modality.

Distinctions Between ML Tasks. In Fig. 1, each application circle is colored depending on the ML task it is primarily formulated as. An ML task is the type of prediction made by the model. The most traditional tasks are *Classification* and *Regression*. *Classification* assigns a category to each input image, while *Regression* predicts a numerical value from their features. Beyond these, a Clustering task groups data instances into clusters based on shared characteristics. When applied to images, Clustering yields a segmentation map, leading us to prefer the term *Segmentation*. *Pixel Classification* also produces a segmentation map by individually processing and categorizing each pixel. However, *Pixel Classification* typically uses pixel-level features (e.g., spectral values, texture) as input, whereas *Segmentation* also considers contextual information and spatial relationships between pixels. Finally, *Object Detection* not only assigns categories but also provides bounding boxes indicating the objects' location, enabling the identification and categorization of multiple objects within a single image. While some applications are closely associated with a specific ML task, others are more versatile and can be formulated in diverse ways. Fig. 1 includes two examples: "Landcover/Land use," formulated five times as a *Pixel Classification* task and four times as a *Classification* task, and "Cloud Coverage Classification," expressed once as a *Segmentation* task. Ultimately, how RS applications are formulated as ML tasks reflects the decisions and priorities of the individual studies.

Converting RS Problems into ML Solutions. *Target Surveillance* is a major focus in the literature. Though uncommon in satellite imagery due to the typical low resolution of the images, this trend might be explained by the proportion of airborne data. Indeed, Unmanned Aerial Vehicles (UAVs) enable better resolution and, subsequently, higher detection accuracy. To mitigate low accuracies, many experiments express *Target Surveillance* applications as scene-wise *Classification*, a solution relying on prudent patchification of the original full-size RS image[8]. Furthermore, we notice a significant focus on "Cloud coverage detection." Such application is motivated by the importance of clouds in optical images. Indeed, throughout the year, clouds cover over 66% of Earth's surface [121], significantly reducing the information obtainable from RGB images[9]. Regarding ML tasks, most surveyed studies frame their applications as supervised *Classification*. In contrast, *Regression* appears only once, making it the most under-represented task. Similarly, most datasets used for *Pixel classification* are labeled and result in supervised problems, except for the "Landcover/Land use" unsupervised classification of Gyaneshwar and Nidamanuri [42]. The *Segmentation* of urban areas follows the same trend, though Ratnakumar and Nanda [99] use unsupervised clustering. Finally, all *Object detection* applications use multi-class datasets and single-shot models to detect multiple instances in a unique pass.

RQ1 TAKEAWAYS

Landscape and environmental analyses are the prominent focus of the surveyed literature (52%). Target surveillance, frequently simplified as scene-wise classification, dominates the rest of the research (45%), with 30% of the applications addressing defense concerns. Regarding data modalities, RGB data is dominant (43%), with HSI remaining niche and SAR primarily used for water environments.

3.3 Diversity and Trends in ML Models (RQ2)

With RQ1, we analyzed the wide spectrum of RS applications and discussed how each experiment translates into an ML task. In RQ2, we study the diversity of models used to interpret this information. Fig. 2 shows the yearly distribution of models from the surveyed studies, categorized by ML task and labeled by each model's "core", i.e., its key architectural

[8]RS images are too large for most ML models to process directly. The process of patchification divides them into smaller, more manageable patches.
[9]Conversely, SAR wavelengths are specifically chosen to be outside the absorption spectrum of clouds, allowing radar signals to pass through without significant attenuation.

characteristics. Although this survey includes studies from 2014 onward, the earliest publication appears in 2017, with significant exploration beginning in 2019. From this point onward, the number of experiments and the diversity of architectures steadily increased, reflecting growing interest in the domain.

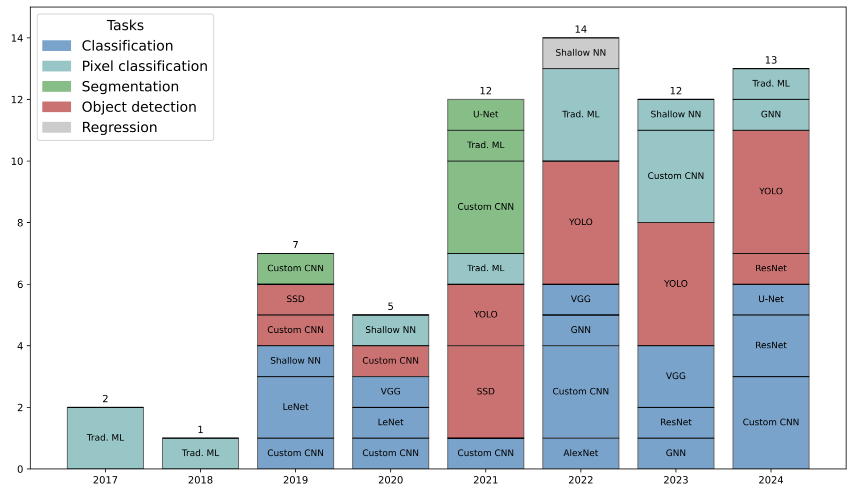

Fig. 2. ML model core architectures grouped per year and colored by their corresponding tasks

Evolution of ML Approaches to RS Problems. Surprisingly, more applications are expressed as *Pixel classification* rather than *Segmentation*, despite the latter being the modern approach for pixel-level analysis [6]. Indeed, *Segmentation* networks, like U-Net [102], have access to spatial information and generally outperform *Pixel classification* methods. However, resource constraints and computation overload also drive the model choice. Unlike deep *Segmentation* architectures, *Pixel classification* experiments often employ shallow networks or traditional ML, which are not only less demanding computationally but also easier to implement and parallelize. This combination of efficiency and simplicity likely explains their prevalence in this FPGA-focused survey, despite the broader RS trend favoring *Segmentation*. The rest of the problems are formulated as *Classification* and *Object detection* tasks. Together, these two categories represent over two-thirds of the experiments. Such a trend can be explained by the versatility of formulating problems as *Classification* tasks, as well as the importance of target monitoring in RS.

Examining Model Core Architectures. Model cores are chosen to highlight the fundamental differences between architectures. In particular, traditional ML methods, such as SVMs [25] or Decision Trees [96], are isolated from Neural Networks (NNs), just as Shallow Networks are distinguished from Deep Neural Networks (DNNs) [68]. We split CNNs—the vast majority of the models analyzed in this survey—between renowned architectures, such as LeNet [69], AlexNet [65], VGG [112], U-Net [102], or ResNet [47], and custom CNN architectures. Notably, "Custom CNN" includes both newly designed CNNs and named architectures used only once in this survey, such as SqueezeNet [54] or ENet [94]. Finally, we classify YOLO (v2, v3, v4) [11, 100, 101] and SSD [77] architectures separately, even though renowned CNN architectures are frequently used as the backbones of detection networks. Indeed, the detection layers added to the feature extraction network (Fully Connected (FC) layers for YOLO and convolution layers for SSD) introduce fundamental differences to the architectures, such as the multiple heads of YOLO networks. Such characteristics reinforce the specialization of these models for object detection tasks.

A CNN-Focused Landscape. Although shallow networks, such as MLPs, and traditional methods, like Decision Trees and SVMs, are featured in some studies [42, 80]—often due to their compatibility with resource-limited hardware—, the vast majority of contributions leverage DL algorithms. Within DL, convolutional architectures are strongly preferred: 76% of all models are CNN-based. Models like AlexNet, ResNet, and VGG are occasionally used, often for benchmarking purposes, but custom-designed CNNs are the most prevalent (27% of all models), reflecting designers' needs for medium-sized architectures typically with 6 to 10 layers. All detection networks are single-shot architectures like YOLO and SSD. Two-stage detection alternatives like R-CNN [35] are not used in the analyzed experiments. Indeed, even if YOLO and SSD architectures are typically larger than R-CNN, the absence of a second pass through the networks highly reduces the computational complexity and processing time, making single-shot networks efficient solutions for resource-limited platforms like FPGAs [143].

Selecting the Right Model. Model selection is influenced by several factors, including the intrinsic suitability of architectures for specific tasks (e.g., YOLO and SSD for object detection) and hardware constraints. For example, while SVMs can fit on most FPGA boards, larger models like ResNet50, with $\sim 25M$ parameters and high memory and computational demands, are unsuitable for smaller devices. This drives the prioritization of lightweight architectures, such as MobileNets [51, 52, 106], and explains the large number of custom CNN designs [75, 143]. Lastly, the software used to integrate and deploy the method on the FPGA platform also drives the model choice. Indeed, if the designers are unfamiliar with low-level FPGA design, only the architectures already supported by higher-level toolchains are available. We further explore this aspect later, through RQ5 in Section 4.2.

RQ2 TAKEAWAYS

With a steadily growing number of publications per year, formulating RS applications as *Classification* problems is prevailing (36%). Of all the models, 76% are convolution-based of which 27% are custom CNNs with an average of 9 layers. All detection models are single-shot networks and 85% of the traditional ML methods are used in *Pixel classification* tasks.

3.4 The Role of the Computing Platform (RQ3)

As explored in the previous sections, deploying AI-supported RS applications in resource-constrained environments requires formulating the application as a suitable ML task and selecting an appropriate model. These decisions are not only deeply intertwined with the selection of the hardware platform, but also driven by project requirements. For example, space deployment introduces harsh environmental conditions, including Single-Event Effects (SEEs)[10] in onboard electronics caused by radiation exposure [67], tight power budgets from 20W to 95W for SmallSats payloads [64], and extreme temperature swings from $-150°$C to $+150°$C [40]—all of which place stringent demands on such edge computing hardware. In light of these challenges, this section explores FPGAs' specific attributes and suitability for deploying AI on the edge compared to other hardware platform alternatives.

FPGA Usage in RS. FPGAs are reconfigurable integrated circuits, that offer design flexibility at the cost of programming complexity. Often available off-the-shelf, FPGAs can operate independently on the edge or connect to other computational platforms via network cables or PCIe. Fig. 3 presents the different FPGAs used in each study. The devices are spread along the x-axis depending on their year of release, while the y-axis represents the number of DSP slices

[10]SEEs manifest as Single-Event Upsets (SEUs)—soft errors altering memory—or as Single-Event Latch-ups (SELs)—potentially destructive hard errors requiring a power reset. Refer to the supplemental material for developed space constraints.

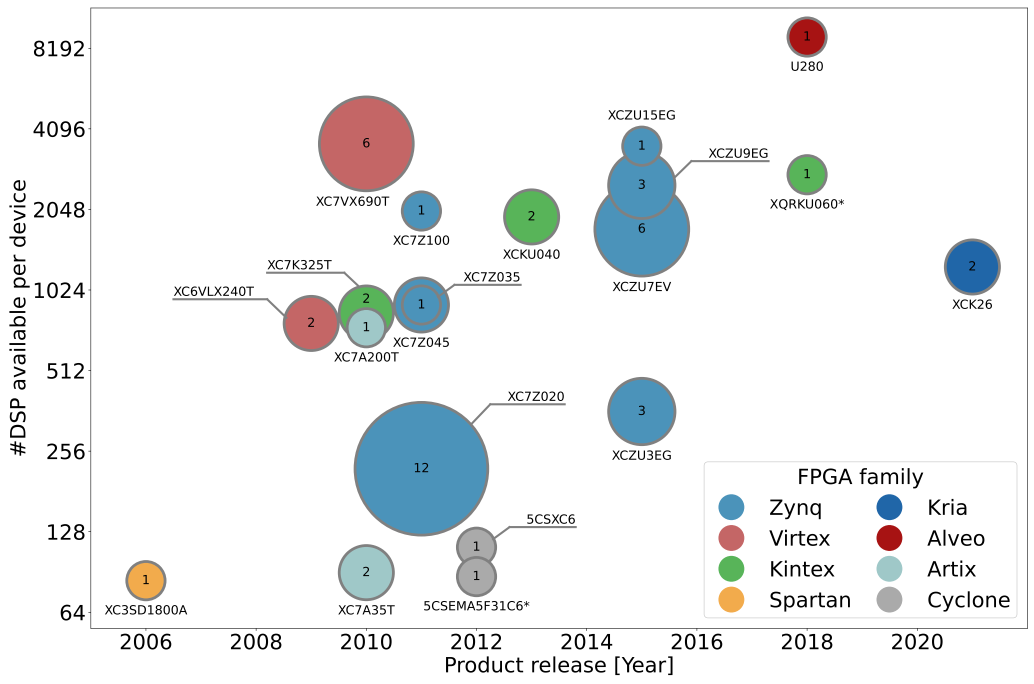

Fig. 3. Distribution of FPGAs used across articles. Unlike other figures/tables, it shows FPGA products used per article (46) instead of per experiment (66). * indicates additional boards used in the studies, absent from the experiments reported in Table 3.

present on the board—relevant for arithmetic computations, like ML. Except for one experiment using the Cyclone V FPGA from Intel, we observe that all considered studies deploy hardware from AMD (formerly Xilinx). The most common family with 60% are Zynq FPGAs, which package FPGA programmable logic with CPUs on a single die to support networking and general-purpose tasks. Such products will be referred to as System-on-Chip (SoC) FPGAs. In contrast, datacenter FPGAs, such as the Alveo Series, are often offered as PCIe cards that can be inserted into a server.

In addition to FPGAs, designers can select from several families of Processing Units (PUs) when deploying AI models on edge devices, each offering unique advantages. In the following subsections, we look at the alternatives: CPUs, GPUs, and ASICs (custom, TPUs, and VPUs). We analyze the authors' motivations to select FPGA over alternatives and mention, when available, the comparisons made by the studies.

FPGAs vs. CPUs. Central Processing Units (CPUs) are ubiquitous general-purpose processing units, widely available as Commercial Off-The-Shelf (COTS) components. While well-suited for complex data transformations and system management, CPUs are limited in parallel processing capabilities, often falling behind specialized hardware in raw computational power. Evaluating FPGA-based designs against CPUs is a common practice in the surveyed studies. These comparisons often involve desktop and server processors, including older Intel i7 models (up to 8th gen) [46, 71, 80, 133], newer i7/i9 models (12th and 13th gen) [53, 138], AMD Ryzen 3990x [137], and Broadwell Xeon server CPUs [89, 127, 141]. While these desktop and server CPUs offer high computational performance, they are not designed for energy-efficient applications, making direct comparisons with FPGA implementations less straightforward. In contrast, some studies include mobile CPUs, such as the AMD 4800H [137, 143], which provide a more power-conscious alternative. Similarly, Huang et al. [53] compare their ETAUS system against an embedded ARM Cortex A530 CPU and achieve significant performance improvements—×25 GOP/s/W and ×37 FPS.

Independently of direct comparison to CPUs, one of the most frequently cited advantages of FPGAs is their superior energy efficiency [8, 18, 42, 48, 71, 76, 82, 131, 133, 140]. For instance, Yang et al. [133] report a power efficiency ratio 29 times superior to an Intel Core i7-8700 K. Indeed, FPGAs' power consumption can be optimized using various design techniques, as detailed by Grover and K.Soni [41]. Furthermore, FPGAs are often favored for their radiation tolerance, with vendors like AMD offering radiation-hardened and radiation-tolerant Virtex and Kintex products. Although similar technologies are available for CPUs and microcontrollers, such as ARM Cortex-R or LEON families, FPGAs' inherent reconfigurability facilitates the implementation of mitigation techniques, like Triple Modular Redundancy (TMR), where critical logic is triplicated and majority voting used to mask errors. On the downside, radiation-hardened FPGAs are significantly more expensive than rad-hard CPUs and microcontrollers. Their SRAM-based memory is also vulnerable to SEUs, requiring mitigation techniques [104]. While investigating SRAM sensibility, Siegle et al. [110] highlight that, although CPUs and microcontrollers remain essential onboard, especially for the OBC, modern payload processing applications increasingly rely on FPGAs. Over time, new solutions to deploy ML algorithms on embedded devices have been developed, shifting the focus from CPUs to domain-specific accelerators [97, 108].

FPGAs vs. GPUs. Graphics Processing Units (GPUs) possess many small cores, enabling high parallelism and performance for tasks involving massive amounts of data and large matrix computations. They are widely used for training NNs. Consequently, many studies compare FPGA-based designs against high-performance NVIDIA GPUs. These include Maxwell [46, 127] and Pascal architectures [75, 89, 131, 139–141], which introduce native half-precision support. Later architectures, such as Volta [71, 84] and Turing [87, 132, 133, 143], contain Tensor Cores, processing units dedicated for Deep Learning operations. Ampere GPUs [136–138] introduce native support for low precision integer operations, thus heavily benefiting from quantization. The latest Blackwell GPUs [91] support precision as low as 4-bit floating-point, decreasing FPGAs' benefits gained from using custom datatypes. GPUs' considerable processing capabilities come at the cost of significant memory and power consumption—often exceeding 300W [22]—making them impractical for most edge deployments. To address these constraints, edge GPUs offer improved efficiency for AI inference. Computing boards, like the NVIDIA Jetson Nano [90], integrate CPUs with energy-efficient GPUs microarchitectures. Yet, Huang et al. [53] report that their FPGA-based system ETAUS achieves 2.7× GOP/s/W compared to Jetson Nano. Similarly, the AP2D-Net presented by Li et al. [76] achieves 2.8× FPS/W compared to the best model of the DAC System Design Contest 2018 [129] deployed on a Jetson TX2. Notably, Li et al. [75] compare their FPGA implementations against the theoretical peak performance of edge GPUs like the Jetson AGX Xavier. They highlight that a significant gap in achievable performance still exists, even after extensive design optimization.

While GPUs deliver high computational throughput, FPGAs typically exhibit better power efficiency in edge scenarios [126]. For instance, Li et al. [71] report achieving 134% of the FPS of a Tesla V100 while consuming 10× less power, whereas Li et al. [75] find that an NVIDIA GTX 1070 Ti outperforms a Zynq-7000 in GOP/s/W efficiency by a factor of three. These variations highlight that power efficiency is highly implementation-dependent. The complex architecture and high transistor density of GPUs pose significant challenges for circuit-level radiation hardening. While the NVIDIA Tegra K1 (TK1) has demonstrated the ability to withstand radiation levels present in short-duration LEO missions [7]. To the author's knowledge, no edge GPU has been explicitly designed for space-grade reliability. The need for ionizing-radiation tolerance to ensure mission reliability is more readily addressed by the accessibility of radiation-tolerant FPGAs [119]. AMD's KCU105 kit, for example, is a commercial equivalent of the space-grade Kintex Ultrascale XQRKU060 used in the VIDEO project—covered by Neris et al. [86]. In summary, while GPUs—especially edge GPUs—offer strong computational performance and power efficiency trade-offs for ML tasks, FPGAs remain the preferred choice in power- and space-constrained environments due to their energy efficiency and radiation tolerance.

FPGAs vs. ASICs (TPUs/VPUs). Application-Specific Integrated Circuits (ASICs) are fully custom-designed chips optimized for specific applications, potentially offering unmatched performance and energy efficiency. Yet, their design is a complex, lengthy, and expensive process—no study has compared an FPGA-based design against a suitable custom ASIC. Many studies list reconfigurability as a key advantage of FPGAs over ASICs [76, 99, 117, 130, 133, 140]; in defense and space for example, re-programmability enhances mission versatility by enabling the "deploy and program" paradigm[11] [48, 71]. Consequently, FPGAs are increasingly replacing traditional ASICs due to their faster time-to-market, cost-efficiency in small-batch applications, and competitive performance [98]. Despite these advantages, ASICs remain the optimal choice for ultra-low-power applications, as the generic nature of FPGAs cannot match the efficiency of fully customized hardware [4].

Tensor Processing Units (TPUs) are commercially available ASICs optimized for Deep Learning, primarily used in data centers [60]. While the Google Coral TPUs [39] target edge devices, no study in this survey directly compared FPGA designs to a TPU. Li et al. [75] only compare their Zynq 7000 design against Google cloud TPU's theoretical peak performance. Vision Processing Units (VPUs), like Intel's Movidius Myriad series, focus on low-power CNN inference. However, compared to the Myriad 2 VPU [132], FPGAs hold a key advantage in radiation tolerance. In a direct comparison, Rapuano et al. [98] build on the CloudScout initiative [36] and conclude that the Myriad 2 VPU is suitable only for short Low Earth Orbit (LEO) missions due to radiation susceptibility—unlike the space-grade Kintex Ultrascale XQRKU060 they also evaluate and is SEL-immune. Performance-wise, their design shows a 2.4× speed-up compared to the VPU, but with 1.8× higher power consumption and a longer development time.

A Large Catalog of Computing Platforms. While many hardware platforms are available to accelerate AI inference on the edge, only a few COTS systems support heavier models like DNNs. Examples include NVIDIA's Jetson Nano GPU [90], Google's Coral TPU [39], and Intel's Myriad 2 VPU [56], all presenting solid alternatives for terrestrial edge computing. However, their susceptibility to radiation limits their use in space applications, where FPGAs offer a more resilient alternative [98]. While floating-point operations on FPGAs may exhibit comparable—or worse, due to lower clock frequency—performance to other embedded platforms, the true advantage of FPGAs emerges when leveraging fixed-point, typically 8-bit, or even binary (1-bit) operations [123]. This capability, often not fully supported or efficiently implemented on other hardware platforms, allows FPGAs to achieve significant performance and efficiency gains.

Through the motivations of the surveyed studies, and in line with the conclusions of Wang et al. [123] and Lentaris et al. [70], a generic guideline for hardware platform selection, considering development effort and performance targets, can be formulated. GPUs, TPUs, and VPUs offer quicker paths to achieving initial performance, especially for computationally complex DNNs [9]. FPGAs, however, promise superior performance and efficiency if more development effort and hardware expertise are invested in custom accelerator design and optimization. For applications demanding ultimate performance and willing to bear the highest development costs and longest design cycles, custom ASICs represent the pinnacle, albeit with reduced flexibility [4]. In the following sections, we look at the technical aspects of the surveyed studies, exploring the methodologies and techniques employed to implement and optimize ML models on FPGAs for RS applications.

[11] Allowing post-launch updates and in-orbit reconfiguration.

RQ3 TAKEAWAYS

AMD FPGAs overwhelmingly dominate the surveyed hardware, comprising nearly all devices (98%). The preferred FPGAs are SoC boards, particularly the Zynq family (57%). Most boards are equipped to run DNNs (61% with > 500 DSPs), and almost all devices have accelerated at least one DL model in an experiment. Reconfigurability, power efficiency, and radiation concerns are the primary motivations for choosing FPGAs over other DNN-friendly COTS accelerators.

4 From AI Models to FPGA Deployment: Design Strategies and Challenges

4.1 Optimizing AI Models for Efficient Inference (RQ4)

Before deploying a model on an FPGA, it typically requires adaptation and compression. These optimization steps are closely linked to hardware design decisions—discussed in RQ5 and RQ6. In RQ4, we introduce key network compression techniques and discuss their impact and requirements. Since traditional ML methods are already lightweight, we primarily discuss optimizations for computationally intensive models, like NNs.

Architectural Optimizations via Lightweight Backbones. Most NNs include a feature extractor, or backbone, responsible for refining and compressing the input's essential information. Extracting rich, high-quality features often relies on large and deep backbones, which come with high computational costs. To lighten these costs, designers can exploit lightweight backbones. For instance, Yang et al. [133] replace YOLOv3's DarkNet53 backbone with GhostNet [44], reducing the number of learnable parameters by a factor of 3. Similarly, Li et al. [71] swap YOLOv4's CSPDarknet53 backbone for MobileNeXt [144], lowering the parameter count from 60.08M to 36.44M, with only a 1.65% mAP drop—compensated via other customizations.

As discussed in RQ2, the existence of many architectures make selection a challenging task. Sabogal and George [104] compare four DL segmentation models, showing that ENet [94] outperforms U-Net while being 23.81× more computationally efficient and requiring 20.56× less memory. Similarly, Yang et al. [132] compare three popular lightweight backbones: MobileNetv1, MobileNetv2, and SqueezeNet [54], concluding that MobileNetv1 achieves the best trade-off for their SAR ship detection application. MobileNetv1 [52] also introduces a width multiplier for layer-wise scaling, leveraged by Suh et al. [113] to explore different deployment scenarios in terms of accuracy and FPGA resource use. Beyond swapping architectures, Kim et al. [63] propose a multi-phase cloud coverage detection pipeline where images first undergo a uniformity check and classification via a small CNN (TriCloudnet) before full segmentation with U-Net. Such workflow avoids the computation of cloud-clear or overcast images, reducing processing time and power consumption by 48.7%. When sacrificing some feature representation capabilities is manageable, lightweight and multi-phase approaches highlight that questioning a model's depth and width can result in significant computational savings.

Reducing the Complexity of Convolutional Layers. Given the ubiquity of convolution operations in Computer Vision, lightweight backbones commonly focus on efficiently computing them. For instance, MobileNets architectures [51, 52, 106] replace standard convolution layers by depth-wise separable convolutions [111]. This form of factorized convolutions separate the standard convolution operation in two stages: filtering, with single 2D filters for each input channel, and combination, by linearly combining the output of the filtering with a 1x1 convolution. When considering an input tensor $h_i \cdot w_i \cdot d_i$, a kernel $k \cdot k \cdot d_i \cdot d_j$, and an output tensor $h_i \cdot w_i \cdot d_j$, a standard convolution costs

$h_i \cdot w_i \cdot d_i \cdot d_j \cdot k \cdot k$, while depth-wise separable convolution costs $h_i \cdot w_i \cdot d_i(k^2 \cdot d_j)$, a computational reduction of a factor $\approx k^2$. For example, Howard et al. [52] use depth-wise separable convolution to reduce computation by 9× for a loss of 1% on ImageNet. As an alternative path to convolution approximation, All Adder Networks [17] replace the multiplications in convolution kernels by additions. While it does not reduce computational complexity, additions are generally less expensive than multiplications. In particular, the All Adder NN of Zhang et al. [139] achieves 3 TOP/s, the highest computational throughput of the survey.

Pruning Techniques for Compact Networks. Pruning reduces computational complexity and memory costs by removing non-essential weights or entire channels from an NN. While the usual overparameterization of NNs aids training, it results in many redundant weights [114]. After training, pruning nullifies the least significant weights (in contribution to the output) to reduce this redundancy, but often requires fine-tuning to recover lost accuracy. The resulting trade-off between parameter reduction and task accuracy can be adjusted with a pruning factor, which determines the percentage of weights to be nullified. To further enhance pruning efficiency, sparsity is often enforced during training via techniques like L1-regularization [137, 143]. For example, Zhang et al. [137] train their GNN with lasso regression—adding an L1-penalty term to the loss—to encourage weight shrinkage and facilitate pruning. Channel pruning removes entire feature channels, enabling more efficient hardware execution. Yang et al. [133] prune Ghost-YOLO channels, reducing the number of parameters from 23.52 M to 6.71 M at a cost of 2.93 % mAP[12]. Similarly, Nguyen et al. [87] prune 60 % of filter weights, incurring only a 0.85 % mAP loss. Beyond weights and channels, certain families of models also have the opportunity to prune input data. For example, Zhang et al. [137] reduce the computational complexity of their GNN by 92.8 % by pruning input graph vertices, causing only a 0.17 % OA drop. To leverage the efficiency improvements reached through pruning, Yang et al. [132] perform hardware-guided progressive pruning. The method integrates an initial coarse pruning, mixed precision Quantization-Aware Training (QAT), resource-per-layer cost estimation, and a final resource-aware fine pruning. Pruning is frequently paired with quantization for further efficiency gains [45, 143].

Custom Datatypes Through Quantization. One of FPGA's key computational advantages over alternative hardware platforms, like GPUs, lies in custom datatype operations. Quantization reduces numerical precision by converting full-precision 32-bit floating-point (*fp32*) parameters into lower-precision formats such as 16-bit fixed-point (*i16*), 8-bit integer (*i8*), or even binary (*b*). Low-bandwidth datatypes enable highly efficient arithmetic operations, particularly on FPGA logic, where binary multiplications can be performed using simple XOR operations. While quantization reduces computational cost and memory footprint, it inevitably perturbs model parameters,[13] shifting them from their convergence point and causing accuracy degradation compared to full-precision implementations [34]. For instance, Sabogal and George [104] report an average accuracy drop of 1% across four segmentation models. Similarly, in object detection, Zhao et al. [143] observe a 0.42% drop in mAP, while Ni et al. [89] report a classification accuracy reduction of 0.03% for VGG16 and 0.06% for ResNet-34. Although the quantization-induced loss rarely exceeds 1%, it can be unacceptable in scenarios requiring high model reliability.

To mitigate accuracy degradation, designers may opt to re-train the network, effectively choosing one of two standard approaches: Post-Training Quantization (PTQ) or Quantization-Aware Training (QAT). PTQ, sometimes called direct quantization, is straightforward to implement as it applies quantization directly after training. However, PTQ's simplicity can cause significant accuracy degradation, especially in models with a wide parameter distribution [58]. For instance,

[12]High accuracy costs are usually mitigated with other optimizations, in this case via Knowledge Distillation (KD) [49], discussed further in Section 6.
[13]Several factors influence the effectiveness of a quantization scheme, such as its uniformity, symmetry, and granularity. For a more detailed discussion on quantization techniques, we recommend the comprehensive survey of Gholami et al. [34].

Wei et al. [126] observe over 50% accuracy loss across all tested bit widths when applying PTQ to a LeNet-5 for MSTAR classification. In contrast, QAT simulates low-precision arithmetic already during training, resulting in better robustness to quantization noise [95], albeit at a higher computational cost [136].

Beyond PTQ and QAT, mixed-precision quantization assigns different bit widths to different sections of the model. This method preserves accuracy by keeping sensitive layers at higher precision while applying aggressive quantization to less critical layers. For example, Yang et al. [132] identify inefficient layers by estimating layer-wise latency and resource consumption. Rapuano et al. [98] take this idea further by manually assigning custom bit widths to each input, filter, and output, achieving a 48% memory footprint reduction for CloudScout. Alternatively, some approaches like Zhang et al. [141] mitigate accuracy loss by selectively retaining *fp32* precision for activations while quantizing only the weights. Similarly, Wei et al. [126] introduce a symmetric QAT method that quantizes weights but preserves normalization and activation in *fp32* format. The resulting computation overhead is motivated by the need to preserve the model's accuracy.

Quantization is indispensable to achieve FPGA designs that are competitive with higher-frequency computing platforms. While PTQ is often sufficient to achieve 8-bit quantization with minimal accuracy degradation [85], QAT is preferable when maintaining accuracy is critical. Mixed-precision quantization further optimizes efficiency by balancing accuracy and computational cost.

Combining Optimization Techniques. All the methods discussed above effectively optimize NNs, but their combined use often yields the best results. For example, Zhao et al. [143] analyze in an ablative fashion the parameters and accuracy reduction of each subsequent technique applied. The resulting figure allows to clearly track which method provides the highest parameter reduction—backbone replacement in their case—and which method helps recover lost accuracy—here, data augmentation. Similarly, Yang et al. [133] follow a structured design flow to progressively refine YOLOv4. They first replace its backbone with GhostNet [44] (leveraging depth-wise separable convolutions), then apply channel pruning to drastically reduce parameters. To restore accuracy, they incorporate Knowledge Distillation (KD) [49], before finally quantizing the model. This multi-step process results in a model with just ≈ 10% of the original parameters and 5% of the memory footprint. A similar approach is presented by Han et al. [45], who combine pruning, quantization, and Huffman coding (weight clustering) to achieve a 35× storage reduction on AlexNet and 49× on VGG16, all without accuracy loss. Ultimately, the effectiveness of these optimization techniques depends not only on model compression but also on how they align with FPGA accelerator design, which we explore next.

RQ4 TAKEAWAYS

To reduce computational load, 26% of DNNs leverage lightweight backbones or efficient architectures, primarily MobileNets (43%). Out of all DL experiments, 91% use quantization, mostly to *int*8 format, and 26% leverage pruning to compress their model. Fine granularity mixed precision quantization presents significant opportunities for further network compression.

4.2 Finding the best framework to efficiently use FPGAs (RQ5)

Implementing NNs on CPUs and GPUs typically involves using stable and mature software frameworks and drivers. In contrast, automatic deployment frameworks for FPGAs have not seen the same level of adoption—most implementations depend on manual frameworks and varying design patterns. To support future work in selecting suitable implementation frameworks and design patterns, we provide a dedicated second taxonomy in Table 3. It presents key metrics relevant to the respective FPGA implementation, organized into three column groups. First, *Implementation Choices* includes the implementation framework *'Impl.'*, the model family *'Fam.'*, the design pattern *'P'* (*S* for Specific, *F* for Flexible), the *'FPGA'* product, and the *'Model Name'*. Second, *Design Metrics* covers the precision post quantitization *'Prec.'*, the model complexity *'C**[OP]**'*, and the memory footprint *'**[MB]**'* (†: model parameters are located on-chip, and not loaded from external memory). Additionally, we report the DSP *'D**[%]**'* and BRAM *'B**[%]**'* resource utilization. Finally, *Performance Metrics* attempts to compare different designs with each other. In particular, we report the operating frequency in *'**[MHz]**'*, computational throughput[14] *'T**[GOP/s]**'*, power consumption *'P**[W]**'*, computational efficiency *'T/P'*, Latency *'L**[s]**'*, and temporal throughput in *'**[FPS]**'*.

Implementation Choices. The designs are split between manual implementations—Hardware Description Language (HDL) and High-Level Synthesis (HLS)—and automatic frameworks—FINN [118], Vitis AI [5], MATLAB, and VGT. HDL is the most commonly used language (≈ 45%), offering precise control of the generated circuit. However, most designs do not exploit this level of control to use all available resources nor push the theoretical limit of the clock frequency. High-Level Synthesis (HLS) tools offer a more abstract approach to FPGA programming as they enable the use of high-level languages to create larger designs with less lines of code. Even with HLS, a solid grasp of FPGA architecture and hardware concepts remains necessary, especially when compared to GPU programming environments, like CUDA. In the reported designs, we detect no overarching trend in worse resource utilization or clock frequency of HLS designs compared to HDL designs. The average resource utilization of implementations are ≈30% DSPs and 50% BRAM for HDL and 46% DSPs and 43% for HLS with a high variance between individual designs. Most manual designs focus on CNN-based networks (≈ 71%), a well-studied network architecture commonly implemented using an array of Processing Elements (PEs) that pass intermediate results between each other.

To reduce the overhead of porting CNNs to FPGAs automatic frameworks have been developed. The two most commonly used in this survey are AMDs Vitis AI and the FINN compiler. The FINN compiler [118] automatically generates a model *Specific* accelerator for CNNs. Once implemented, the FINN design is solely useful for the specific network. It heavily depends on quantization to reduce the network complexity and memory footprint. Myojin et al. [84] binarize a 4 layers CNN, and Pitonak et al. [95] compare 2,3 and 4-bit quantization using FINN. The Vitis AI compiler [5] supports the *Flexible* design approach, using a Deep Learning Processor Unit (DPU) flashed to the FPGA able to run multiple different AI workloads. The peak Operations Per Cycle (OPC) of the DPU can be customized and matched to the algorithm [5]. Sabogal and George [104] test multiple networks with different DPU configurations, demonstrating that some networks have utilization as low as 14.2% (ESPNet) and up to 85.6% (U-Net) using the B1024 (1024 OPC) architecture. Vitis AI can leverage multiple DPU cores to increase performance as demonstrated by Nguyen et al. [87] and Yu et al. [136]. On the downside, Vitis AI requires PTQ or QAT, as it only supports the *i8* datatype [5]. Other automated frameworks include MATLAB (2023a) [31] and the Xilinx System Generator (XSG) [42], but both works do not go into detail concerning their implementation. Bahl et al. [8] use VGT, a VHDL compiler generating circuits for abstract CNNs similar to FINN; it has, however, no continuing support. Frameworks that are not classified do not provide enough information on their implementation.

Design Pattern. We split each workload into two distinct design patterns. *Flexible (F)* designs are bitstreams that can support multiple different kernels and network topologies without requiring reconfiguration. Supporting different networks and kernels usually requires weights to be loaded from off-chip memory. Zhang et al. [137] use a *Flexible* design

[14]See Section 6.3 for a discussion about the confusion between **[GOP]**, **[GOPs]**, and **[GOP/s]**.

Table 3. FPGA Optimization Taxonomy Table

	Implementation Choices						Design Metrics					Peformance Metrics					
	Impl.	**Fam.**	**P**	**Ref.**	**FPGA**	**Model Name**	**Prec.**	**C[OP]**	**MB**	**D[%]**	**B[%]**	**MHz**	**T[GOP/s]**	**P[W]**	**T/P**	**L[s]**	**FPS**
Manual	HDL	CNN	S	[55]	7Z045	HO-ShipNet	i16	-	-	44	44	-	-	1.9	-	1.0e-3	-
				[98]	ZU7EV	CloudScout	i?	-	13.3	67	46	115	-	3.4	-	1.4e-1	-
				[121]	7A35T	CNN	f32*	-	-†	-	-	36	1.1	0.1	9.8	-	-
				[133]	7Z020	Ghost-YOLOS	i16	8.41G	12.6	58	60	150	29.5	3.0	9.9	3.2e-1	-
				[141]	7VX690T	Q-IORN	i8	-	121.5	21	27	200	209.6	6.3	33.2	1.5e-1	6.8
				[139]	7VX690T	A2NN	i8,f32	14.9G	-†	9	86	200	3047	8.3	368.4	4.9e-3	203.2
			F	[48]	ZU15EG	2D CNN	i8	684 K	1.2	5	71	100	7.1	8.4	0.8	9.7e-5*	-
					ZU15EG	3D CNN	i8	4.23M	0.1	9	76	100	3.8	8.4	0.5	1.1e-3*	-
					ZU15EG	HybridSN	i8	101 M	20.5	37	80	100	13.2	8.4	1.6	7.7e-3*	-
				[75]	7Z100	CBFF-SSD	i16	-	-	57	60	200	452.8	19.5	23.2	4.3e-2	-
				[76]	ZU3EG	AP2D-Net	i(mix)	-	-	80	75	300	130.2	5.6	23.3	3.3e-2	30.5
				[89]	7VX690T	Impr. YOLOv2	i8,i32	379 G	49.4	23	54	200	387	15.0	25.9	9.8e-1	-
					7VX690T	ResNet-34	i8,i32	7.33G	21.3	23	54	200	182	15.0	12.2	4.0e-2	-
					7VX690T	VGG16	i8,i32	30.6G	14.7	23	54	200	344	15.0	23.0	8.9e-2	-
				[126]	7K325T	LeNet-5 f32	f32	-	6.6	34	33	100	-	-	-	2.3e-3	-
					7K325T	LeNet-5 i8	i8	-	1.7	14	16	100	-	-	-	2.3e-3	-
				[131]	7A200T	Impr. VGG16	i8	40.9G	14.8	12	29	200	23.1	3.4	6.8	1.8e+0	-
					7A200T	Impr. YOLOv2	i8	379 G	49.4	12	29	200	22.2	3.4	6.5	1.7e+1	-
				[140]	7Z035	Impr. YOLOv2	i8,f32	379 G	-	21	74	200	111.5	6.0	18.7	3.4e+0	-
		ML	S	[28]	7Z020	Decision Tree	i8	-	-†	10	86	-	-	3.5	-	3.0e-1*	-
				[80]	7Z020	SVM	f32*	-	-†	38	-	200	-	-	-	1.1e-4*	-
				[82]	3SD1800A	MLP	i8,i32	-	-†	26	-	-	-	-	-	-	-
				[99]	6VLX240T	RDBC	-	-	-†	-	-	230	-	0.1	-	2.9e-2	-
				[109]	7VX690T	LPDBL	f32*	-	-†	14	15	200	-	-	-	-	-
				[130]	6VLX240T	BRAM_DSP	i8	-	-†	91	52	192	-	4.8	-	2.1e-5	-
					6VLX240T	LUT_MUL	i8	-	-†	0	0	189	-	4.5	-	2.1e-5	-
		GNN	S	[16]	7VX690T	SAM-GNN	f32*	-	-	13	14	200	-	-	-	-	-
	HLS	CNN	S	[63]	7Z020	TriCloudNet + U-Net	-	-	3.6	49	60	-	-	0.2	-	1.0e+0	1
				[124]	7Z020	CNN	b	-	-†	11	100	-	69.8	2.4	29.3	-	7
				[132]	7VX690T	CNN2@0.7	i?	450 M	0.1†	64	21	250	298	5.8	51.4	7.0e-4	652
					7VX690T	CNN4@0.7	i?	560 M	0.1†	69	21	250	366	6.6	55.5	7.0e-4	636
					7VX690T	CNN6@1.6	i?	600 M	0.1†	76	18	250	235	5.1	46.1	1.6e-3	384
			F	[127]	ZU9EG	Impr. YOLOv3	i8	-	8.5	-	-	-	-	-	-	7.0e-3	-
			-	[86]	KU040	AlexNetLite	i16	5.71G	-?	31	39	200	-	-	-	-	-
					KU040	MobileNetv1Lite	i16	50.0M	-?	31	39	200	-	-	-	-	-
		ML	S	[13]	ZU7EV	Deep Belief Network	f32	-	-†	10	0	250	-	-	-	5.4e-3	-
		GNN	F	[138]	U280	GNN	f32*	-	-	43	37	260	400	38.0	10.5	2.7e-2	759
				[137]	ZU7EV	GNN	f32	2.13M	1.0†	74	91	125	-	6.3	-	1.1e-4	9500
Automatic	FINN	CNN	S	[46]	7Z020	CNN	b	-	-†	-	-	-	-	-	-	3.3e-1	-
				[84]	7Z045	BNN	b	-	-†	-	-	-	-	-	-	2.1e-2	-
				[95]	7Z020	CloudSatNet-1 Q2	b	-	1.4†	14	-	100	-	2.5	-	-	15.5
					7Z020	CloudSatNet-1 Q4	i4	-	3.1†	14	-	100	-	2.6	-	-	15.5
	Vitis AI	CNN	F	[53]	ZU19EG	ETAUS	i8	-	-	-	-	106	329.9	1.6	199.9	1.1e-1	9.3
				[71]	ZU7EV	RFA-YOLO	i8	-	24.8	-	-	200	-	15.8	-	-	28
				[87]	ZU7EV	YOLOv4	i8	105 G	245.0	-	-	100	-	30.1	-	1.1e-1	17.9
					ZU7EV	YOLOv4-tiny 3L	i8	15.3G	24.1	-	-	100	-	26.4	-	1.9e-2	125
				[104]	ZU3EG	ENet	i8	4.06G	0.4	70	75	300	-	3.4	-	-	25.2
					ZU3EG	ESPNet	i8	3.71G	0.3	70	75	300	-	3.1	-	-	11.7
					ZU3EG	FPN	i8	17.3G	5.8	70	75	300	-	4.0	-	-	14.1
					ZU3EG	U-Net	i8	96.6G	7.4	70	75	300	-	3.4	-	-	2.7
				[119]	ZU9EG	ResNet-50	i8	-	-	-	-	-	-	-	-	1.1e-2	93
					ZU9EG	SICNet	i8	-	-	-	-	-	-	-	-	1.8e-3	530.7
				[136]	ZU7EV	Impr. YOLOv4-tiny	i8	-	-	-	-	300	-	20.0	-	-	295.9
				[143]	K26	YOLOv4-MobileNetv3	i8	-	5.7	59	-	200	-	7.2	-	-	48.1
	MATLAB	CNN	S	[31]	ZU9EG	ResNet-18+YOLOv2	i8	-	-	-	-	250	-	-	-	-	46
		ML	S	[42]	7A35T	CAG-SC2S	i32	-	-?	93	21	70	-	0.3	-	5.3e-6*	-
					7A35T	CAL-SC2S	i32	-	-?	96	25	65	-	0.4	-	5.7e-6*	-
					7A35T	SVM	i32	-	-?	49	6	72	-	0.2	-	5.1e-6*	-
	VGT	CNN	S	[8]	5CSXC6	C-FCN++	i16	4.70K	0.0†	-	-	100	-	-	-	1.5e-1	7
Unclassified	-	CNN	S	[18]	7K325T	LeNet-5	f32	-	-	-	33	100	-	-	-	2.3e-3	-
			F	[113]	ZU3EG	SSD 0.25x	i8	4.04G	-	73	47	200	138	2.4	57.5	-	34.2
					ZU3EG	SSD 0.5x	i8	15.6G	-	73	55	200	150	2.6	57.7	-	9.6
					ZU3EG	SSD 1.0x	i8	61.6G	-	-	-	200	158	2.0	79.0	-	2.6
			-	[125]	7Z020	Lightweight CNN	i8	4.60G	-	70	25	100	-	3.3	-	-	60
		ML	S	[43]	7Z020	MLP	f32*	-	-†	26	1	-	-	0.1	-	-	-
			-	[117]	7Z020	WNS	f32*	-	-†	-	-	-	-	-	-	-	-

FPGA: AMD/XILINX FPGA names are stripped of leading XC to increase readability
Mem: † Uses On-Chip Memory, **?** Memory Location unknown
Latency: * Latency of a single Pixel
Model Name: **RDBC** Roller Dung Beetle Clustering, **WNS** Weightless Neural System

with on-chip memory, pre-loading memory to the weight buffer, and amortizing the access over multiple classifications. The support for different layers comes at a small cost in computational performance, but allows implementing larger models that would otherwise not fit on the FPGA using a *Specific* approach. The average footprint of the 20 *Flexible* designs is 25*MB* compared to 15*MB* for the 10 *Specific* designs. Unlike *Flexible* designs, *Specific* designs only support a single network architecture. Deploying a different model requires re-implementation of the bitstream or is not supported. This allows the design to be hyper-specialized to the model, making full use of the customizability of the FPGA. *Specific* designs can use different datapaths for different pruning and quantization of individual layers. Differentiating model families (*'Fam.'*) is relevant as different ML models have vastly distinct requirements. For example, CNNs spend most of their time performing convolution, a task with high computational intensity [98], whereas GNNs require a high memory bandwidth [138]. The FPGA device also influences the performance, similar to CPUs, the number of transistors increases over the generations, enabling designers to use more resources such as DSPs, BRAM, and LUTs (see 3.4).

Design Metrics. The *'Prec.'* column reports the precision of operations post-quantization. The most popular quantization is *i8*, with 51% of all implementations and 44% of manual ones using either *i8* or a mixture of *i8* with *f32* or *i32*. Next, we report the model complexity *'C*[**OP**]*'*, an essential metric as a higher complexity requires more computational resources to produce the same temporal throughput *'FPS'*. In addition, the relationship between model complexity and memory footprint can indicate whether a model is compute- or memory-bound. Most of the designs do not reach the compute boundary nor provide an off-chip memory bandwidth analysis making a further discussion impossible.

The next metric is the model footprint and the location of parameters. FPGAs, similarly to CPUs, contain restricted on-chip memory, e.g., < 5MB for Zynq MPSoC chips, and a main off-chip memory (512MB to 32GB). Using on-chip memory eliminates most memory bandwidth concerns as distributed BRAM provides very high bandwidth. On-chip memory weights can either be stored in BRAM—pre-loaded before execution—or stored inside LUTs—disabling any change in parameters. *Specific* designs mostly use on-chip memory with only 25% using off-chip memory, while 93% of *Flexible* designs use off-chip memory. In particular, the FINN compiler stores weights on-chip, such as BRAM and LUTs, while Vitis AI loads weights from external memory. Storing the weights on-chip leads to the design being mostly limited by BRAM [28, 124, 139]. The average memory footprint of on-chip implementation is only 0.83*MB* compared to 28*MB* for off-chip memory implementation.

The last two middle columns present the DSP & BRAM utilization. A high utilization can indicate a good selection of the model for the FPGA, while a low utilization can indicate bad scalability of the design. This excludes network architectures that cannot use DSPs and BRAMs efficiently, such as BNNs—using LUTs for most operations—or GNNs—with low data locality and computational intensity. Yahiaoui et al. [130] compare an implementation using only LUTs with a DSP- and BRAM-based implementation. They show that the BRAM_DSP implementation can double the number of pixel processed in parallel, while using the same amount of resources.

Performance Metrics. Comparing different models on CPUs and GPUs is already difficult; for FPGAs, where the design can vary even more, finding a fair comparison is especially complex. One common metric relevant for comparison is the computational throughput, marking the peak number of operations per second. Nevertheless, operations with different precisions cannot be directly compared, as floating-point computations are more expensive than integer operations, and smaller bit widths yield higher **[GOP/s]**. For example, the highest throughput is reported as 3 TOP/s [139]. However, the solution uses an All Adder NN, which approximates the functionality of a convolution filter by using only additions. Moreover, the intermediate feature maps resulting from the convolution layers are sent back to the CPU. Thus, the Fully-Connected (FC) layers, corresponding to 92% of the parameters, are executed on the CPU, drastically reducing the memory footprint. The remaining layers only represent $10M$ parameters, which can fit on the boards 6.6MB BRAM due to the quantization to 4-bit. Using BRAM eliminates any memory bottlenecks and the main computation, a 4-bit adder, can be implemented using solely LUTs, resulting in the extreme reported throughput. The second highest reported throughput is 452 GOP/s [75]. Compared to Zhang et al. [139] the design uses 16-bit values loaded from memory, and the main operations are Multiply-Accumulate (MAC), counted as two operations. The DSP block inside a 7-series Xilinx FPGA can perform one MAC operation per cycle and they manage to keep 1152 DSPs (60%) busy achieving $1.97 \times 1152 \times 200MHz \approx 452GOP/s$. Sabogal and George [104] and Suh et al. [113] use double datarate DSPs to increase the performance—an optional optimization inside Vitis AI. Doubling the clock frequency of DSP components, which can be clocked higher than the rest of the design, roughly doubles the available computational capability.

A majority of experiments (≈ 72%) report the power consumed by the FPGA, a value pointless on its own as the total energy required to compute a sample is more important. To attempt a more informative comparison between different models and implementations, we calculate the computational efficiency *'T/P'* in $[\frac{GOP/s}{W}]$. Zhang et al. [139] achieve the highest computational efficiency $368\frac{GOP/s}{W}$ due to the high throughput detailed previously. On the other hand, He et al. [48] have an extremely low $< 2\ \frac{GOP/s}{W}$ efficiency, even though they use a low-precision datatype (*i8*). Another key design metric is the frequency *'MHz'*, which depends on the speed-grade and technology of the device, as well as the critical paths in the design. A low frequency, compared to designs on the same FPGA, can hint at a poorly designed circuit with long critical paths. *'Latency'* and *'FPS'* values are model- and implementation-specific, and uncomparable between different workloads and implementations. Nevertheless, we include them in Table 3 to ensure completeness. It is important to separate *'Latency'* from *'FPS'*, as, generally, multiple inputs can be processed inside the accelerator pipeline increasing the FPS significantly compared to a non-pipelined implementation that can process one element at a time.

RQ5 TAKEAWAYS

Extensive work in implementing CNNs on FPGAs has produced various automated frameworks supporting common layers. Within the surveyed studies we could not find a trend showing better performance of manual designs in any of the metrics. *Specific* design patterns are commonly used (≈ 75%) with limited on-chip memory, while *Flexible* design patterns tend to support larger models (+ 10 MB in average).

4.3 Optimizing Performance for FPGA implementations (RQ6)

FPGAs have a reconfigurable architecture, which enables designers to implement energy-efficient dataflow architectures that re-use data, and are adapted to the compute operations of the algorithm. However, FPGAs' reconfigurability comes at the cost of frequency—compared to CPUs and GPUs. Therefore, the use of different kinds of parallelism is necessary to achieve similar performance.

Instruction Level Parallelism (ILP) exploits the execution of consecutive instructions in parallel, for example overlapping load and MAC operations [113]. This level of parallelism is exploited in almost every design and can be performed manually in HDL or automatically using HLS control directives for loop statements. Consecutive iterations of a loop can overlap, increasing the throughput and amortizing the latency over multiple loop iterations. In addition, ILP designs commonly exploit Data Level Parallelism (DLP), for example by executing multiple MAC operations in parallel to perform convolution [131].

After optimizing individual tasks using ILP, designers can leverage Task Level Parallelism (TLP), for example, by overlapping different layers in a NN—a mechanic heavily exploited by FINN [118]. The compiler connects individual layers with First-In First-Out (FIFO) streams and starts executing a layer as soon as its first inputs are produced by the previous layer. Increasing the batch size increases TLP, enabling different inputs to be processed at the same time. The disadvantage of larger batch sizes is a possible increase in latency, leading to most implementations using a batch size of 1 during inference. Increasing computation puts pressure on the memory bandwidth, and most designs use multiple PEs in a Systolic Array [66] to reduce off-chip memory accesses. Furthermore, multiple PEs enable simple scaling to use most of the FPGA resources; this approach can be found in many designs [16, 48, 75, 95, 98, 109, 113, 124, 126, 127, 130–133, 137–141].

Using multiple levels of parallelism, most designs can be scaled to reach the computational limit for floating-point operations on FPGAs, which require multiple DSPs per operation. To increase the **OP/s** available, designers can choose to reduce datatypes precision, hence reducing the number of resources required per operation. Most designs use quantization to a fixed-point datatype to increase the number of available operations. In addition to DSPs, which can perform basic arithmetic operations on integers, LUTs can perform any 6-bit to 1-bit mapping or a 5-bit to 2-bit operation. Thus, a BNN or low-precision network can be mapped to LUTs directly, allowing for high peak **OP/s** [139]. The trade-off between LUT and DSP utilization is automatically explored by Yang et al. [132]. In addition to computational benefits, lower precision reduces off-chip memory requirements and can allow the network to fit entirely into BRAM. Another option to increase available operations is to run DSPs at double the frequency, effectively doubling their throughput [104, 113]. For networks with low computational intensity, the focus is placed on creating an advanced network to reduce the number of access to off-chip memory as much as possible [137, 138].

RQ6 TAKEAWAYS

Most manual designs use a combination of different forms of parallelism and quantization to increase the throughput of the design. To reduce the off-chip memory bandwidth, designs can re-use data, commonly employing a Systolic Array of PEs. Comparing the quality of different designs proves difficult, and most surveyed studies do not compare their CNN implementations with previous work.

5 AI and FPGA Synergies: towards Next-Gen Earth Observation Missions

5.1 Cutting-Edge AI-FPGA Synergies for RS Applications (RQ7)

While the setup of each study differs—making any comparison challenging and generalization even harder—we would like to highlight and discuss some of the best solutions surveyed.[15]

Balancing Accuracy and Computation. Deploying large ML models on FPGAs often requires balancing task accuracy and hardware performance. In this regard, Ni et al. [89] achieve competitive Overall Accuracy (OA) on the NWPU-RESISC45 dataset [20] with both ResNet-34 (93%) and VGG16 (92%) deployed on their *Flexible* accelerator design. The authors leverage mixed-precision datatypes and multiples PEs to reach high throughput while achieving accuracies competitive with the State Of The Art (SOTA) of 96%. On DOTAv1.0 [128], Zhang et al. [140] implement a lightweight version of YOLOv2, achieving 67% mAP. Although the accuracy remains significantly below SOTA (82%), their compressed YOLOv2 model enables on-the-edge computations. As a general note, comparing SOTA-driven studies

[15]Dataset accuracies were compared to the benchmarks available at https://paperswithcode.com/sota.

with HW-aware FPGA implementations is challenging without a cost/benefit analysis. Quantifying the computational cost of a 1% accuracy gain would clarify the trade-off between task performance and hardware efficiency, yet, this is often missing in SOTA literature.

Compact and Fast ML Models. Encoding models directly on-chip avoids loading parameters from peripherals, granting significant speed-up, as demonstrated by the OSCAR-RT framework in Yang et al. [132]. By mapping all CNN layers onto on-chip FPGA resources and using mixed-width quantization and optimized pipeline scheduling, they consistently compress YOLOv2 with different lightweight backbones under 0.5MB while maintaining accuracies above 90% mAP. In particular, their MobileNetv1 [52] and MobileNetv2 [106] designs run in 0.7ms, the lowest latency of the surveyed DL models. Wei et al. [126] achieve LeNet-5 inference in 2.29ms via a custom QAT algorithm, though their ablation study hints at sub-optimal FPGA resource usage. Upadhyay et al. [119] classify cloud coverages with their SICNet CNN in 1.8 ms, the lowest latency of Vitis AI implementations. The lowest per-pixel latency in this survey is achieved by the 2D CNN implemented by He et al. [48], reaching 97μs on a Zynq US+. However, such performance is facilitated by the abundance of resources of the Zynq US+ series, which comes with a power consumption of 8.4W.

Low-Power Implementations for Resource-Constrained Systems. As RQ3 highlights, power efficiency is the main concern for hardware designs, especially onboard SmallSats [64]. Traditional ML methods and shallow networks offer low power consumption, such as MLP [43], RDBC [99], Fuzzy ARTMAP [130], and SVM-based models [42] consuming less than 1W. When it comes to DL, the CNN HO-ShipNet of Ieracitano et al. [55] consumes 1.90W on a Zynq 7045, while the Ghost-YOLOS of Yang et al. [133] uses 2.98W. Suh et al. [113] achieve particularly impressive results using their custom 8-bit UniPOT quantization scheme and making use of DSP double rate, reaching 2.4, 2.6, and 2.0W for their 3 SSD sizes. Among the solutions implemented with Vitis AI, Huang et al. [53] reports the lowest power consumption, 1.65W, a noteworthy result for a ResNet-50-based system like their ETAUS.

Lightweight Models and Quantization with Compensation. The combination of lightweight backbones and fixed-point quantization is a recurring and effective strategy in the surveyed studies [63, 71, 75, 86, 87, 132, 133, 136, 143]. These two optimization methods are broadly applicable in RS, except in extreme scenarios where traditional ML methods are preferred to DNNs. Lightweight backbone replacement or direct use of an efficient architecture, like ENet [94], or a *"tiny"* adaptation, for example, YOLOv4-tiny [87, 136], are used in ≈ 20% of the surveyed experiments. To compensate for accuracy drops, designers typically enhance the baseline architecture. For example, Nguyen et al. [87] add a third YOLOv4 head to detect targets at different spatial scales and optimize training with data augmentation strategies to achieve higher baseline accuracies. Following such enhancements, techniques like pruning or quantization can further reduce the solution's computational load.

RQ7 TAKEAWAYS

Accuracy loss due to design decisions during FPGA porting remains a challenge in achieving state-of-the-art results. While real-time performance is frequently achieved, most models still require significant power (42% > 5W). Lightweight architectures combined with moderate (> 4-bit) quantization offer a promising trade-off between accuracy and computational efficiency.

5.2 Advancing Onboard Processing: ML Approaches for Satellites and UAVs (RQ8)

One primary motivation for this survey was to understand how AI methods currently onboard spacecraft are deployed.

Onboard PhiSat-1: Pioneering Work in AI for EO. PhiSat-1[16] is a frequently cited mission [37] which deployed the first CNN model performing onboard cloud coverage classification: CloudScout [36]. ML-based methods accelerate data processing pipelines and recently became excellent at several essential tasks [115]. This is particularly true in RS, where the exponentially growing amount of data has outpaced the capacity for manual expert labeling. Deploying DL models directly onboard relieves downlink constraints and reduces the need for on-ground pre-processing. This saves time, conserves resources, and simplifies dataflows. For example, CloudScout classifies the cloud coverage of freshly acquired images, discarding overly cloudy images that contain little valuable EO information. Beyond hardware constraints, like limited memory and power, using a CNN for such a critical task implies a responsibility that comes with additional restrictions. As such, CloudScout has to ensure a False Positive rate under 1%, to avoid discarding appropriate images. In a follow-up study, Rapuano et al. [98] explore the possibility of using an FPGA instead of the original Myriad 2 VPU. They conclude that, although FPGA-based designs take longer to develop and consume more power, their superior customizability, lower latency, and radiation hardness make them suitable for long-term LEO missions. In a recent work, Cratere et al. [26] provide a comprehensive overview of the projects building on the CloudScout initiative, comparing all related cloud detection studies.

Onboard OPS-SAT: Experimental AI Deployment in Space. A second ESA mission is particularly relevant to onboard ML for RS: OPS-SAT[17]. This flying laboratory featured an Altera Cyclone V SoC and was designed for experimental ML deployments. The C-FCN++ model presented by Bahl et al. [8] was deployed onboard OPS-SAT. With 273 parameters (47KB), C-FCN++ is the only architecture explored in the study able to fit on-chip the Altera Cyclone V SoC. It uses dilated convolutions to expand early receptive fields and improve performance without increasing parameter count. Ultimately, C-FCN++ could perform cloud segmentation on a full-scale RS image in 150ms, less than required for the camera of OPS-SAT to acquire the next image, granting real-time performance. For further investigations of ML onboard OPS-SAT, see Kacker et al. [61], who tested several algorithms to prepare for the mission Beaver-Cube-2[18].

Preparing Deep Learning for Spacecraft. Beyond deployed solutions, several studies contribute to preparing DL models for onboard spacecraft processing. In particular, Kim et al. [63] prepared for the A-HiREV mission[19] and present a cloud classification pipeline with goals similar to CloudScout. Their multi-phase approach uses three stages: a uniformity check, a lightweight ternary classifier called TriCloudNet, and a pruned U-Net. Each stage filters out extreme cases, reducing the workload for subsequent, more computationally complex models. This image prioritization method reduces downlink data by 40–50% and cuts processing time and power consumption by around 50%. Designed as a direct comparison to CloudScout, CloudSatNet-1 [95] classifies cloud coverage to relieve downlink, with a strong focus on low-power consumption and a minimized False Positive rate. More recently, Upadhyay et al. [119] develop SICNet, a cloud detection CNN optimized for real-time performance. The designers focus on deployment ease using Vitis AI and prepare further applications like fire and object detection. Zhang et al. [139] also target downlink reduction with A2NN, a model for direct onboard classification of RS images. As for solutions monitoring ships, Yang et al. [132] present OSCAR-RT, an end-to-end algorithm/hardware codesign framework dedicated to real-time onboard ship detection with SAR images. Their three experiments considerably focus on real-time performance using on-chip implementations. Neris et al. [86] use lightweight CNNs to identify ships and airplanes, deploying them on a Kintex US FPGA, which has a radiation-hardened counterpart, the XQRKU060—also used by Rapuano et al. [98]. Ieracitano et al. [55] also

[16] 6U CubeSat, launched 09/2020, https://www.esa.int/Applications/Observing_the_Earth/Ph-sat
[17] 3U CubeSat, 12/2019 - 05/2024, https://www.esa.int/Enabling_Support/Operations/OPS-SAT
[18] 3U CubeSat educational mission from MIT, expected launch 2025, https://www.nanosats.eu/sat/beavercube-2
[19] 6U CubeSat platform from the Korea Aerospace Research Institute (KARI)

target ship monitoring with HO-ShipNet, a CNN enhanced with xAI techniques for increased transparency, achieving 95% accuracy. As discussed in Section 3.4, radiation resilience remains a key requirement for space deployment [110]. Sabogal and George [104] study SEEs effects on DL applications, quantifying performance degradation in Xilinx DPU implementations [5]. Such effects are mitigated through circuit-level hardening or software solutions, like Triple Modular Redundancy (TMR). By contrast, UAV platforms operate in less radiation-exposed environments, enabling more flexible trade-offs in model design and hardware selection.

AI-Accelerated UAVs Payloads: Autonomous and Real-Time Insights. Spacecraft are not the only edge RS platforms benefiting from ML; numerous studies explore ML-supported pipelines for UAVs. Wang and Qiu [124] deploy a CNN on a Zynq-7000 to detect objects in images from the 2018 DAC System Design Contest, targeting real-time inference under low-power constraints. Suh et al. [113] design three sizes of SSD models to detect drones achieving high energy efficiency on a Zynq US+. Addressing a practical use case, Yu et al. [136] develop an Improved YOLOv4-tiny to detect abnormal railway track fasteners. Implemented with Vitis AI, their solution achieves $\approx$ 300 FPS and 95.1% mAP, demonstrating performance levels suitable for onboard UAV deployment. Focusing on UAV navigation, Fraczek et al. [28] experiment with Decision Trees and Support Vector Machines to classify terrain and support automated landing of UAVs. Similarly, Wang et al. [125] merge optical and MilliMeter-Wave (MMW) radar data to quickly detect UAV obstacles. Their lightweight CNN reaches real-time performance with 60 FPS and low-power 3.3W on a Zynq-7020. By far the most complete surveyed UAV-focused study, Huang et al. [53] present the ETAUS system, a custom UAV based on the Pixhawk 4 drone to monitor Taiwan's Air Quality Index (AQI). ETAUS is powered by a CNN achieving high accuracy for AQI classification and cryptographic modules implemented on the FPGA logic. ETAUS also uses a YOLOv4 pre-trained from Vitis AI Model Zoo to detect—and later blur—private and sensitive information, such as license plates[20]. The study focuses on maintaining high accuracy for a reliable solution and achieving sufficient FPS for real-time performance.

Onboard Challenges: Real-Time, Memory, and Power Constraints. Real-time processing remains the central concern across the surveyed studies, as keeping pace with the continuous stream of newly acquired images is essential for operational success. To meet this demand, designers minimize model computations and maximize execution parallelism. In some contexts, such as the PhiSat-1 payload and CloudScout [36], strict memory footprint limitations introduce additional constraints. Low power consumption is also a recurring objective, though the diversity of mission requirements makes it difficult to extract consistent trends. For instance, Nguyen et al. [87] could afford to onboard a YOLOv4-tiny 3L with a 26.4W power budget on their UAV.

RQ8 TAKEAWAYS

Although only three studies explicitly target space missions, the widespread focus on onboard deployment suggests the broad applicability of the research to future onboard systems. Space missions are often geared towards surveillance, while UAVs address more localized applications.

[20]This section of the algorithm was not detailed in the study and is therefore not reported in this survey.

6 The Missing Pieces: Gaps and Opportunities in FPGA-enabled ML for RS

This section highlights key research gaps and opportunities of this emerging field. Following PRISMA 2020 guidelines [93], we first outline the limitations of our survey, then present promising study directions, and conclude with recommendations for future work.

6.1 Survey Limitations

While Scopus[21] and Web of Science (WoS)[22] are the two most comprehensive public databases [15], we limited our search to WoS to maintain focus and feasibility; a preliminary analysis showed its journal coverage better aligns with the scope of this review. Besides, as with all systematic surveys, the search results inherently depend on the chosen query terms. This can lead to omissions, such as works deploying on FPGAs but using broader description like 'resource-limited hardware' in their abstracts. Similarly, although we aimed for comprehensive ML-related queries, some lesser-known models may have been overlooked.

While the heterogeneity in the surveyed literature contributes to a broad and extensive body of work, it complicates direct comparisons of experiments. Comparing FPGA implementations is particularly challenging, as different designs use distinct networks and devices. Furthermore, most CNN-based studies do not benchmark against prior FPGA-based CNNs implementations, limiting our ability to identify commonly compared architectures. Lastly, distinguishing between *Flexible* and *Specific* accelerators often relies on our interpretation of the described designs.

6.2 Research Gaps Analysis

In this section, we identify critical research gaps from the surveyed literature. To offer insights to guide future research directions, we structure the gaps into five subtopics.

Incomplete Spectrum of RS Use Cases. RQ1 shows that the surveyed applications do not cover the full spectrum of RS problems. While edge AI does not resonate well with unstressed applications that can be processed on the ground, like long-term climate monitoring, other onboard-relevant use cases remain unexplored, such as data compression and disaster response. Indeed, the NewSpace era enables real-time alerts for numerous emergency situations, such as wildfires. Similarly, local authorities greatly benefit from post-disaster damage assessments, e.g., after floods [81]. Exploring lightweight implementations of SOTA methods on FPGA could significantly advance both research and industry adoption. Concurrently, DL-based data compression pipelines have gained relevance over the past decade, including for EO data [38]. With SmallSats imaging payloads generating data at ≈ 640 Mbps [119], efficient onboard compression methods present a valuable research opportunity.

Overlooked Prevalent DL Architectures. Although RQ2 highlights the vast diversity of encountered models, the surveyed literature overlooks several widespread model families. Indeed, all DL-based studies rely on CNN or GNN architectures, with no examples of transformers or recurrent networks. This gap may partly reflect the recent emergence of Vision Transformers (ViTs), which have not yet seen mature FPGA implementations. However, it may also highlight the relative implementation simplicity of graph flows and convolution pipelines on FPGA platforms. Nevertheless, ViTs have significantly advanced DL in RS research [115] and, despite their complexity, their superior performance makes them a promising target for edge deployment [105]. Similarly, no surveyed work explores RNNs, despite their natural fit for temporal tasks—recurrent in RS, such as monitoring change across time or along flight paths. Investigating the

[21] https://www.scopus.com/
[22] https://www.webofscience.com/

application of RNNs onboard flying platforms represents a valuable research direction. At the same time, the absence of support for transformer and recurrent architectures in FPGA toolchains (e.g., FINN, Vitis AI) poses a barrier to broader adoption. To unlock the potential of modern DL architectures in edge environments, automated frameworks must evolve to support these emerging layers, including their quantized and pruned variants.

Beyond FPGAs: Coarse-Grained Reconfigurable Arrays (CGRA). While RQ3 reveals a wide diversity of FPGAs, especially high-end products suited for DL inference, we fail to find ML solutions on CGRAs. Unlike FPGAs, CGRAs harden the commonly used arithmetic units, improving energy efficiency and allowing higher unit density. This reduces configurability, but offers enhanced performance. As discussed in Section 4.2, CNNs are often implemented with Processing Elements (PEs) organized in a systolic array. The AMD Versal[23] platform features a systolic array of small PEs (AI Engines) capable of basic vector operation. Notably, the Versal AI Edge family combines FPGA programmable logic with CGRA AI engines for on-the-edge DNNs inference. Given the availability of space-grade Versal devices, a detailed comparison between CGRA-based and FPGA-only solutions–particularly for highly quantized models—emerges as a concrete and promising research direction.

Further Refining NNs for Efficient Inference. RQ4 reveals that some widely used compression methods in Computer Vision remain underexplored in the surveyed records. In particular, Knowledge Distillation (KD) appears in only one study [133]. KD is a training strategy where a large 'teacher' model with high learning capabilities transfers its learned representations to a smaller 'student' model [49]. In the on-the-edge context, the student network can be tailored to the target FPGA, maximizing resource utilization and performance. KD can also guide weight pruning to improve compression while limiting accuracy loss [3]. This synergy holds strong potential for deployment in constrained environments. Although a few studies adopt hardware-aware design methods, hardware insights could benefit many optimization techniques. Notably, no surveyed work explores mixed-precision quantization driven by layer-wise hardware feedback, as in Yao et al. [135], who improve energy efficiency and latency by assigning optimal bitwidths per layer. Similarly, we see potential in using AutoML, particularly Neural Architecture Search (NAS), to identify DNN architectures tailored to hardware constraints. Approaches like EfficientNet [116] or MobiletNetv3 [51] are designed using platform-aware NAS, but hardware-aware NAS (HW-NAS) is still a young field with significant room for growth [10]. As GPUs increasingly integrate dedicated CNN units, such techniques become even more relevant for FPGAs, which must exploit their configurability to remain competitive. FPGA-aware NAS offers a promising path to design performant models well-matched to FPGA deployment, a research direction supported by datasets like HW-NAS-Bench [72] that estimate FPGA performance.

Trust in and Interpretability of Onboard AI. Another underexplored area in this survey is Uncertainty Quantification (UQ). UQ methods estimate the confidence of model predictions to improve interpretability and support more reliable decision-making onboard. Only Myojin et al. [84] apply UQ techniques, using Monte Carlo (MC) Dropout [29] to estimate model uncertainty through multiple stochastic forward passes. Other established methods remain unexploited in this context and deserve further investigation; for a comprehensive overview of UQ in Deep Learning, we refer readers to Gawlikowski et al. [32]. Practical benefits of UQ include uncertainty rejection, where predictions exceeding a confidence threshold are discarded, potentially easing downlink stress and increasing reliability. Similarly, explainable AI (xAI) is crucial for critical decision-making onboard, such as alert triggering or data prioritization. Yet, only Ieracitano et al.

[23]https://www.amd.com/en/products/adaptive-socs-and-fpgas/versal.html

[55] employ xAI tools to interpret model decisions. We believe that lightweight xAI and UQ methods can significantly increase the trustworthiness and adoption of AI solutions for onboard processing [50].

6.3 Recommendations for Future Work

Remote Sensing Good Practices. We encourage researchers to use open-source data or to publish their datasets. When a full release is not possible due to confidentiality agreements, studies should detail, as rigorously as possible, sensor specifications, selected scenes, ground truth, data splits, etc. Such transparency is crucial for future readers to fully comprehend the problem and replicate results.

Comprehensive Reporting of Metrics. A significant limitation across many reviewed studies is the lack of thorough metric reporting, which prevents comprehensive evaluation and comparison. Fundamental performance indicators are often incomplete. For example, object detection works may report **mAP**, but omit **mIoU**. Similarly, hardware results may mention throughput and power, but use inconsistent bit-widths (e.g., *i4* vs. *fp32*), skewing energy efficiency comparisons. Even more critical, memory usage is rarely analyzed in depth. While some papers mention memory footprint, very few examine off-chip memory transfers—a common bottleneck for FPGA designs [2, 23]. We also want to highlight the recurrent confusion around the terms **GOP**, **GOPs**, and **GOP/s**. We recommend using **GOP** to denote computational complexity and **GOP/s** for computational throughput. Similarly, the term "throughput" is often used without clarification, though it may refer to either computational throughput [**GOP/s**] or frame throughput [**FPS**].

Micro-architecture of Convolution Kernels. Despite the ubiquity of convolution operations in the surveyed applications, most HDL and HLS studies re-implement convolution kernels from scratch, often without an in-depth study or comparison to established designs. This is surprising given the rich body of prior work on optimized CNN accelerators for FPGAs [1, 2, 118, 122], as well as mature ASIC tape-outs from both academia [19] and industry [59]. Rather than reinventing the convolution micro-architecture, new studies could build on these foundations and shift focus toward application-specific challenges. Furthermore, we have observed that many studies highlight low FPGA resource utilization as a positive outcome, however, it may also indicate poor scalability. Ideally, FPGA resources should be fully utilized, except when limited by external factors like power budgets or memory bandwidth.

Code Availability. Among the analyzed articles, only few provide accessible code repositories. While we recognize the difficulty of sharing runnable HDL environments, recent tools like Vitis AI [5]—which runs in containers and through customized Python scripts—facilitate code sharing. We strongly encourage future authors to share their code and scripts, as they provide valuable information for the readers and increase the long-term impact of their work.

7 Conclusion

The integration of Machine Learning (ML) into Remote Sensing (RS) pipelines is reshaping Earth Observation, enabling real-time, autonomous data processing onboard satellites and UAVs. Facing the growing demands of NewSpace and the need for efficient ML accelerators, FPGAs stand out as the standard payload for SmallSats due to their adaptability, potential for computational speedup, and energy efficiency. This article followed PRISMA 2020 guidelines to systematically review the literature on ML models implemented on FPGA-based platforms for RS applications. We proposed two taxonomies to classify existing research and answered eight research questions to map the literature landscape, unveil design paradigms, and discuss technological synergies. Our analysis highlighted the dominance of CNN-based architectures, the prevalence of AMD FPGAs, and the necessary synergy of quantization with FPGA programmable logic. Despite progress in the field, our findings reveal gaps in research, including limited diversity in RS tasks, underexplored model compression techniques, like knowledge distillation, and the scarcity of uncertainty quantification and explainability methods in FPGA-based ML. Addressing these gaps will be crucial for advancing reliable and efficient onboard AI systems. We hope this survey serves as a valuable resource for researchers and stimulates future advancements in FPGA-enabled Machine Learning for Remote Sensing.

Acknowledgments

This work is supported by the Helmholtz Association under the joint research school "Munich School for Data Science – MUDS" and the German Federal Ministry of Research, Technology and Space, under the project BB-KI Chips, contract no. 16DHBKI020. This work benefited from the use of ChatGPT-4o, which provided alternative formulations of some parts of the manuscript, for example, section titles and the abstract.

References

[1] Mohamed S. Abdelfattah, David Han, Andrew Bitar, Roberto DiCecco, Shane OConnell, Nitika Shanker, Joseph Chu, Ian Prins, Joshua Fender, Andrew C. Ling, and Gordon R. Chiu. 2018. DLA: Compiler and FPGA Overlay for Neural Network Inference Acceleration. https://doi.org/10.48550/arXiv.1807.06434 arXiv:1807.06434 [cs]

[2] Kamel Abdelouahab, Maxime Pelcat, Jocelyn Serot, and François Berry. 2018. Accelerating CNN Inference on FPGAs: A Survey. https://doi.org/10.48550/arXiv.1806.01683 arXiv:1806.01683 [cs]

[3] Nima Aghli and Eraldo Ribeiro. 2021. Combining Weight Pruning and Knowledge Distillation for CNN Compression. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 3191–3198.

[4] Amara Amara, Frédéric Amiel, and Thomas Ea. 2006. FPGA vs. ASIC for Low Power Applications. *Microelectronics Journal* 37, 8 (Aug. 2006), 669–677. https://doi.org/10.1016/j.mejo.2005.11.003

[5] AMD. 2019. *AMD Vitis™ AI Software*. Technical Report.

[6] Saeid Asgari Taghanaki, Kumar Abhishek, Joseph Paul Cohen, Julien Cohen-Adad, and Ghassan Hamarneh. 2021. Deep Semantic Segmentation of Natural and Medical Images: A Review. *Artificial Intelligence Review* 54, 1 (Jan. 2021), 137–178. https://doi.org/10.1007/s10462-020-09854-1

[7] Jose M. Badia, German Leon, Jose A. Belloch, Almudena Lindoso, Mario Garcia-Valderas, Yolanda Morilla, and Luis Entrena. 2022. Reliability Evaluation of LU Decomposition on GPU-Accelerated System-on-Chip Under Proton Irradiation. *IEEE Transactions on Nuclear Science* 69, 7 (July 2022), 1467–1474. https://doi.org/10.1109/TNS.2022.3155820

[8] Gaetan Bahl, Lionel Daniel, Matthieu Moretti, and Florent Lafarge. 2019. Low-Power Neural Networks for Semantic Segmentation of Satellite Images. In *2019 Ieee/Cvf International Conference on Computer Vision Workshops (Iccvw) (IEEE International Conference on Computer Vision Workshops)*. 2469–2476. https://doi.org/10.1109/ICCVW.2019.00302

[9] Robert Bayer, Julian Priest, and Pınar Tözün. 2023. Reaching the Edge of the Edge: Image Analysis in Space. https://doi.org/10.48550/arXiv.2301.04954 arXiv:2301.04954 [cs]

[10] Hadjer Benmeziane, Kaoutar El Maghraoui, Hamza Ouarnoughi, Smail Niar, Martin Wistuba, and Naigang Wang. 2021. A Comprehensive Survey on Hardware-Aware Neural Architecture Search. arXiv:2101.09336 [cs]

[11] Alexey Bochkovskiy, Chien-Yao Wang, and Hong-Yuan Mark Liao. 2020. YOLOv4: Optimal Speed and Accuracy of Object Detection. https://doi.org/10.48550/arXiv.2004.10934 arXiv:2004.10934 [cs]

[12] Mustapha Bouhali, Farid Shamani, Zine Elabadine Dahmane, Abdelkader Belaidi, and Jari Nurmi. 2017. FPGA Applications in Unmanned Aerial Vehicles - A Review. In *Applied Reconfigurable Computing*, Stephan Wong, Antonio Carlos Beck, Koen Bertels, and Luigi Carro (Eds.). Springer International Publishing, Cham, 217–228. https://doi.org/10.1007/978-3-319-56258-2_19

[13] Samuel Boyle, Aksel Gunderson, and Milica Orlandic. 2023. High-Level FPGA Design of Deep Learning Hyperspectral Anomaly Detection. In *2023 Ieee Nordic Circuits and Systems Conference, Norcas*. https://doi.org/10.1109/NorCAS58970.2023.10305467

[14] James B. Campbell and Randolph H. Wynne. 2011. *Introduction to Remote Sensing*. Guilford press.

[15] Arezoo Aghaei Chadegani, Hadi Salehi, Melor Md Yunus, Hadi Farhadi, Masood Fooladi, Maryam Farhadi, and Nader Ale Ebrahim. 2013. A Comparison between Two Main Academic Literature Collections: Web of Science and Scopus Databases. *Asian Social Science* 9, 5 (April 2013), p18. https://doi.org/10.5539/ass.v9n5p18

[16] C Chellaswamy, MM Manjula, B Ramasubramanian, and A Sriram. 2024. FPGA-based Remote Target Classification in Hyperspectral Imaging Using Multi-Graph Neural Network. *MICROPROCESSORS AND MICROSYSTEMS* 105 (March 2024). https://doi.org/10.1016/j.micpro.2024.105008

[17] Hanting Chen, Yunhe Wang, Chunjing Xu, Boxin Shi, Chao Xu, Qi Tian, and Chang Xu. 2020. AdderNet: Do We Really Need Multiplications in Deep Learning?. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 1468–1477.

[18] L Chen, X Wei, WC Liu, H Chen, and L Chen. 2020. Hardware Implementation of Convolutional Neural Network-Based Remote Sensing Image Classification Method. In *Beijing Institute of Technology*, Q Liang, X Liu, Z Na, W Wang, J Mu, and B Zhang (Eds.), Vol. 516. 140–148. https://doi.org/10.1007/978-981-13-6504-1_19

[19] Yu-Hsin Chen, Tien-Ju Yang, Joel Emer, and Vivienne Sze. 2019. Eyeriss v2: A Flexible Accelerator for Emerging Deep Neural Networks on Mobile Devices. https://doi.org/10.48550/arXiv.1807.07928 arXiv:1807.07928 [cs]

[20] Gong Cheng, Junwei Han, and Xiaoqiang Lu. 2017. Remote Sensing Image Scene Classification: Benchmark and State of the Art. *Proc. IEEE* 105, 10 (Oct. 2017), 1865–1883. https://doi.org/10.1109/JPROC.2017.2675998

[21] Gong Cheng, Junwei Han, Peicheng Zhou, and Lei Guo. 2014. Multi-Class Geospatial Object Detection and Geographic Image Classification Based on Collection of Part Detectors. *ISPRS Journal of Photogrammetry and Remote Sensing* 98 (Dec. 2014), 119–132. https://doi.org/10.1016/j.isprsjprs.2014.10.002

[22] C. Collange, David Defour, and Arnaud Tisserand. 2009. Power Consumption of GPUs from a Software Perspective. In *Computational Science – ICCS 2009*, Gabrielle Allen, Jaroslaw Nabrzyski, Edward Seidel, Geert Dick van Albada, Jack Dongarra, and Peter M. A. Sloot (Eds.). Springer, Berlin, Heidelberg, 914–923. https://doi.org/10.1007/978-3-642-01970-8_92

[23] Jason Cong, Zhenman Fang, Michael Lo, Hanrui Wang, Jingxian Xu, and Shaochong Zhang. 2018. Understanding Performance Differences of FPGAs and GPUs. In *2018 IEEE 26th Annual International Symposium on Field-Programmable Custom Computing Machines (FCCM)*. IEEE, Boulder, CO, USA, 93–96. https://doi.org/10.1109/FCCM.2018.00023

[24] Corporation for Digital Scholarship. 2024. Zotero.

[25] Corinna Cortes and Vladimir Vapnik. 1995. Support-Vector Networks. *Machine Learning* 20, 3 (Sept. 1995), 273–297. https://doi.org/10.1007/BF00994018

[26] Angela Cratere, M. Salim Farissi, Andrea Carbone, Marcello Asciolla, Maria Rizzi, Francesco Dell'Olio, Augusto Nascetti, and Dario Spiller. 2025. Efficient FPGA-Accelerated Convolutional Neural Networks for Cloud Detection on CubeSats. *IEEE Journal on Miniaturization for Air and Space Systems* (2025), 1–1. https://doi.org/10.1109/JMASS.2025.3533018

[27] Steve Foga, Pat L. Scaramuzza, Song Guo, Zhe Zhu, Ronald D. Dilley, Tim Beckmann, Gail L. Schmidt, John L. Dwyer, M. Joseph Hughes, and Brady Laue. 2017. Cloud Detection Algorithm Comparison and Validation for Operational Landsat Data Products. *Remote Sensing of Environment* 194 (June 2017), 379–390. https://doi.org/10.1016/j.rse.2017.03.026

[28] P Fraczek, A Mora, and T Kryjak. 2018. Embedded Vision System for Automated Drone Landing Site Detection. In *AGH University of Krakow*, LJ Chmielewski, R Kozera, A Orlowski, K Wojciechowski, AM Bruckstein, and N Petkov (Eds.), Vol. 11114. 397–409. https://doi.org/10.1007/978-3-030-00692-1_35

[29] Yarin Gal and Zoubin Ghahramani. 2016. Dropout as a Bayesian Approximation: Representing Model Uncertainty in Deep Learning. In *Proceedings of The 33rd International Conference on Machine Learning*. PMLR, 1050–1059.

[30] Antonio-Javier Gallego, Antonio Pertusa, and Pablo Gil. 2018. Automatic Ship Classification from Optical Aerial Images with Convolutional Neural Networks. *Remote Sensing* 10, 4 (April 2018), 511. https://doi.org/10.3390/rs10040511

[31] D Garg, D Shah, V Chintapalli, S Paul, and AB Mishra. 2024. Aircraft Detection in Satellite Imagery Based on Deep Learning/AI Techniques and FPGA /SoC Based Hardware Implementation. In *2024 IEEE SPACE, AEROSPACE AND DEFENCE CONFERENCE, SPACE 2024*. 112–115. https://doi.org/10.1109/SPACE63117.2024.10667719

[32] Jakob Gawlikowski, Cedrique Rovile Njieutcheu Tassi, Mohsin Ali, Jongseok Lee, Matthias Humt, Jianxiang Feng, Anna Kruspe, Rudolph Triebel, Peter Jung, Ribana Roscher, Muhammad Shahzad, Wen Yang, Richard Bamler, and Xiao Xiang Zhu. 2023. A Survey of Uncertainty in Deep Neural Networks. *Artificial Intelligence Review* 56, 1 (Oct. 2023), 1513–1589. https://doi.org/10.1007/s10462-023-10562-9

[33] Alan D. George and Christopher M. Wilson. 2018. Onboard Processing With Hybrid and Reconfigurable Computing on Small Satellites. *Proc. IEEE* 106, 3 (March 2018), 458–470. https://doi.org/10.1109/JPROC.2018.2802438

[34] Amir Gholami, Sehoon Kim, Zhen Dong, Zhewei Yao, Michael W. Mahoney, and Kurt Keutzer. 2021. A Survey of Quantization Methods for Efficient Neural Network Inference. https://doi.org/10.48550/arXiv.2103.13630 arXiv:2103.13630 [cs]

[35] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. 2014. Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation. https://doi.org/10.48550/arXiv.1311.2524 arXiv:1311.2524 [cs]
[36] Gianluca Giuffrida, Lorenzo Diana, Francesco de Gioia, Gionata Benelli, Gabriele Meoni, Massimiliano Donati, and Luca Fanucci. 2020. CloudScout: A Deep Neural Network for On-Board Cloud Detection on Hyperspectral Images. *Remote Sensing* 12, 14 (Jan. 2020), 2205. https://doi.org/10.3390/rs12142205
[37] Gianluca Giuffrida, Luca Fanucci, Gabriele Meoni, Matej Batič, Léonie Buckley, Aubrey Dunne, Chris van Dijk, Marco Esposito, John Hefele, Nathan Vercruyssen, Gianluca Furano, Massimiliano Pastena, and Josef Aschbacher. 2022. The Φ-Sat-1 Mission: The First On-Board Deep Neural Network Demonstrator for Satellite Earth Observation. *IEEE Transactions on Geoscience and Remote Sensing* 60 (2022), 1–14. https://doi.org/10.1109/TGRS.2021.3125567
[38] Carlos Gomes, Isabelle Wittmann, Damien Robert, Johannes Jakubik, Tim Reichelt, Michele Martone, Stefano Maurogiovanni, Rikard Vinge, Jonas Hurst, Erik Scheurer, Rocco Sedona, Thomas Brunschwiler, Stefan Kesselheim, Matej Batic, Philip Stier, Jan Dirk Wegner, Gabriele Cavallaro, Edzer Pebesma, Michael Marszalek, Miguel A. Belenguer-Plomer, Kennedy Adriko, Paolo Fraccaro, Romeo Kienzler, Rania Briq, Sabrina Benassou, Michele Lazzarini, and Conrad M. Albrecht. 2025. Lossy Neural Compression for Geospatial Analytics: A Review. https://doi.org/10.48550/arXiv.2503.01505 arXiv:2503.01505 [eess]
[39] Google. 2019. Coral TPU Dev Board. https://coral.ai/docs/dev-board/datasheet/#system-components.
[40] P. Gordo, T. Frederico, R. Melicio, S. Duzellier, and A. Amorim. 2020. System for Space Materials Evaluation in LEO Environment. *Advances in Space Research* 66, 2 (July 2020), 307–320. https://doi.org/10.1016/j.asr.2020.03.024
[41] Naresh Grover and M. K.Soni. 2012. Reduction of Power Consumption in FPGAs - An Overview. *International Journal of Information Engineering and Electronic Business* 4, 5 (Oct. 2012), 50–69. https://doi.org/10.5815/ijieeb.2012.05.07
[42] Dubacharla Gyaneshwar and Rama Rao Nidamanuri. 2022. A Real-Time SC¡SUP¿2¡/SUP¿S-based Open-Set Recognition in Remote Sensing Imagery. *JOURNAL OF REAL-TIME IMAGE PROCESSING* 19, 5 (Oct. 2022), 867–880. https://doi.org/10.1007/s11554-022-01226-y
[43] Bilal Hammoud, Charbel Bou Maroun, Jonas Ney, and Norbert Wehn. 2022. Artificial Neural Networks-Based Radar Remote Sensing to Estimate Geographical Information during Oil-Spills. In *2022 30th European Signal Processing Conference (EUSIPCO)*. 1801–1805. https://doi.org/10.23919/EUSIPCO55093.2022.9909727
[44] Kai Han, Yunhe Wang, Qi Tian, Jianyuan Guo, Chunjing Xu, and Chang Xu. 2020. GhostNet: More Features From Cheap Operations. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 1580–1589.
[45] Song Han, Huizi Mao, and William J. Dally. 2016. Deep Compression: Compressing Deep Neural Networks with Pruning, Trained Quantization and Huffman Coding. https://doi.org/10.48550/arXiv.1510.00149 arXiv:1510.00149 [cs]
[46] Shintaro Hashimoto, Yohei Sugimoto, Ko Hamamoto, and Naoki Ishihama. 2019. Ship Classification from SAR Images Based on Deep Learning. In *Intelligent Systems and Applications, Vol 1 (Advances in Intelligent Systems and Computing, Vol. 868)*, K Arai, S Kapoor, and R Bhatia (Eds.). 18–34. https://doi.org/10.1007/978-3-030-01054-6_2
[47] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep Residual Learning for Image Recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 770–778.
[48] Wenjing He, Yuesong Yang, Shaohui Mei, Jian Hu, Wanqiu Xu, and Shiqi Hao. 2023. Configurable 2D–3D CNNs Accelerator for FPGA-Based Hyperspectral Imagery Classification. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing* 16 (2023), 9406–9421. https://doi.org/10.1109/JSTARS.2023.3321965
[49] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. 2015. Distilling the Knowledge in a Neural Network. https://doi.org/10.48550/arXiv.1503.02531 arXiv:1503.02531 [stat]
[50] Adrian Höhl, Ivica Obadic, Miguel Ángel Fernández Torres, Hiba Najjar, Dario Oliveira, Zeynep Akata, Andreas Dengel, and Xiao Xiang Zhu. 2024. Opening the Black-Box: A Systematic Review on Explainable AI in Remote Sensing. arXiv:2402.13791 [cs]
[51] Andrew Howard, Mark Sandler, Grace Chu, Liang-Chieh Chen, Bo Chen, Mingxing Tan, Weijun Wang, Yukun Zhu, Ruoming Pang, Vijay Vasudevan, Quoc V. Le, and Hartwig Adam. 2019. Searching for MobileNetV3. https://doi.org/10.48550/arXiv.1905.02244 arXiv:1905.02244
[52] Andrew G. Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. 2017. MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications. https://doi.org/10.48550/arXiv.1704.04861 arXiv:1704.04861 [cs]
[53] CH Huang, WT Chen, YC Chang, and KT Wu. 2024. An Edge and Trustworthy AI UAV System With Self-Adaptivity and Hyperspectral Imaging for Air Quality Monitoring. *IEEE INTERNET OF THINGS JOURNAL* 11, 20 (Oct. 2024), 32572–32584. https://doi.org/10.1109/JIOT.2024.3422470
[54] Forrest N. Iandola, Song Han, Matthew W. Moskewicz, Khalid Ashraf, William J. Dally, and Kurt Keutzer. 2016. SqueezeNet: AlexNet-level Accuracy with 50x Fewer Parameters and <0.5MB Model Size. https://doi.org/10.48550/arXiv.1602.07360 arXiv:1602.07360
[55] Cosimo Ieracitano, Nadia Mammone, Fanny Spagnolo, Fabio Frustaci, Stefania Perri, Pasquale Corsonello, and Francesco C. Morabito. 2024. An Explainable Embedded Neural System for On-Board Ship Detection from Optical Satellite Imagery. *ENGINEERING APPLICATIONS OF ARTIFICIAL INTELLIGENCE* 133, 108517 (July 2024). https://doi.org/10.1016/j.engappai.2024.108517
[56] Intel. 2016. Intel® Movidius™ Myriad™ 2 Vision Processing Unit 1GB - Product Specifications. https://www.intel.com/content/www/us/en/products/sku/204771/intel-movidius-myriad-2-vision-processing-unit-1gb/specifications.html.
[57] ISPRS. 2022. 2D Semantic Labeling Contest - Potsdam. https://www.isprs.org/education/benchmarks/UrbanSemLab/2d-sem-label-potsdam.aspx.
[58] Benoit Jacob, Skirmantas Kligys, Bo Chen, Menglong Zhu, Matthew Tang, Andrew Howard, Hartwig Adam, and Dmitry Kalenichenko. 2018. Quantization and Training of Neural Networks for Efficient Integer-Arithmetic-Only Inference. In *2018 IEEE/CVF Conference on Computer Vision*

and Pattern Recognition. IEEE, Salt Lake City, UT, 2704–2713. https://doi.org/10.1109/CVPR.2018.00286

[59] Norman P. Jouppi, Doe Hyun Yoon, Matthew Ashcraft, Mark Gottscho, Thomas B. Jablin, George Kurian, James Laudon, Sheng Li, Peter Ma, Xiaoyu Ma, Thomas Norrie, Nishant Patil, Sushma Prasad, Cliff Young, Zongwei Zhou, and David Patterson. 2021. Ten Lessons From Three Generations Shaped Google's TPUv4i : Industrial Product. In *2021 ACM/IEEE 48th Annual International Symposium on Computer Architecture (ISCA)*. IEEE, Valencia, Spain, 1–14. https://doi.org/10.1109/ISCA52012.2021.00010

[60] Norman P. Jouppi, Cliff Young, Nishant Patil, David Patterson, Gaurav Agrawal, Raminder Bajwa, Sarah Bates, Suresh Bhatia, Nan Boden, Al Borchers, Rick Boyle, Pierre-luc Cantin, Clifford Chao, Chris Clark, Jeremy Coriell, Mike Daley, Matt Dau, Jeffrey Dean, Ben Gelb, Tara Vazir Ghaemmaghami, Rajendra Gottipati, William Gulland, Robert Hagmann, C. Richard Ho, Doug Hogberg, John Hu, Robert Hundt, Dan Hurt, Julian Ibarz, Aaron Jaffey, Alek Jaworski, Alexander Kaplan, Harshit Khaitan, Daniel Killebrew, Andy Koch, Naveen Kumar, Steve Lacy, James Laudon, James Law, Diemthu Le, Chris Leary, Zhuyuan Liu, Kyle Lucke, Alan Lundin, Gordon MacKean, Adriana Maggiore, Maire Mahony, Kieran Miller, Rahul Nagarajan, Ravi Narayanaswami, Ray Ni, Kathy Nix, Thomas Norrie, Mark Omernick, Narayana Penukonda, Andy Phelps, Jonathan Ross, Matt Ross, Amir Salek, Emad Samadiani, Chris Severn, Gregory Sizikov, Matthew Snelham, Jed Souter, Dan Steinberg, Andy Swing, Mercedes Tan, Gregory Thorson, Bo Tian, Horia Toma, Erick Tuttle, Vijay Vasudevan, Richard Walter, Walter Wang, Eric Wilcox, and Doe Hyun Yoon. 2017. In-Datacenter Performance Analysis of a Tensor Processing Unit. In *Proceedings of the 44th Annual International Symposium on Computer Architecture (ISCA '17)*. Association for Computing Machinery, New York, NY, USA, 1–12. https://doi.org/10.1145/3079856.3080246

[61] Shreeyam Kacker, Alex Meredith, Kerri Cahoy, and Georges Labrèche. 2022. Machine Learning Image Processing Algorithms Onboard OPS-SAT. *Small Satellite Conference* (Aug. 2022).

[62] Xudong Kang. 2017. Airport-Beach-Urban Website. Retrieved 2024-11-08 from http://xudongkang.weebly.com/data-sets.html

[63] JH Kim, Y Kim, DH Cho, and SM Kim. 2024. On-Orbit AI: Cloud Detection Technique for Resource-Limited Nanosatellite. *INTERNATIONAL JOURNAL OF AERONAUTICAL AND SPACE SCIENCES* (Dec. 2024). https://doi.org/10.1007/s42405-024-00865-8

[64] Vivek Kothari, Edgar Liberis, and Nicholas D. Lane. 2020. The Final Frontier: Deep Learning in Space. In *Proceedings of the 21st International Workshop on Mobile Computing Systems and Applications*. ACM, Austin TX USA, 45–49. https://doi.org/10.1145/3376897.3377864

[65] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. 2012. ImageNet Classification with Deep Convolutional Neural Networks. In *Advances in Neural Information Processing Systems*, Vol. 25. Curran Associates, Inc.

[66] Kung. 1982. Why systolic architectures? *Computer* 15, 1 (1982), 37–46. https://doi.org/10.1109/MC.1982.1653825

[67] Kevin Lange, Federico Fontana, Francesco Rossi, Mattia Varile, and Giovanni Apruzzese. 2024. Machine Learning in Space: Surveying the Robustness of on-Board ML Models to Radiation. https://doi.org/10.48550/arXiv.2405.02642 arXiv:2405.02642

[68] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. 2015. Deep Learning. *Nature* 521, 7553 (May 2015), 436–444. https://doi.org/10.1038/nature14539

[69] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. 1998. Gradient-Based Learning Applied to Document Recognition. *Proc. IEEE* 86, 11 (Nov. 1998), 2278–2324. https://doi.org/10.1109/5.726791

[70] George Lentaris, Konstantinos Maragos, Ioannis Stratakos, Lazaros Papadopoulos, Odysseas Papanikolaou, Dimitrios Soudris, Manolis Lourakis, Xenophon Zabulis, David Gonzalez-Arjona, and Gianluca Furano. 2018. High-Performance Embedded Computing in Space: Evaluation of Platforms for Vision-Based Navigation. *Journal of Aerospace Information Systems* 15, 4 (2018), 178–192. https://doi.org/10.2514/1.I010555

[71] Chao Li, Rui Xu, Yong Lv, Yonghui Zhao, and Weipeng Jing. 2023. Edge Real-Time Object Detection and DPU-based Hardware Implementation for Optical Remote Sensing Images. *REMOTE SENSING* 15, 3975 (Aug. 2023). https://doi.org/10.3390/rs15163975

[72] Chaojian Li, Zhongzhi Yu, Yonggan Fu, Yongan Zhang, Yang Zhao, Haoran You, Qixuan Yu, Yue Wang, and Yingyan Lin. 2021. HW-NAS-Bench:Hardware-Aware Neural Architecture Search Benchmark. https://doi.org/10.48550/arXiv.2103.10584 arXiv:2103.10584 [cs]

[73] Jianwei Li, Changwen Qu, and Jiaqi Shao. 2017. Ship Detection in SAR Images Based on an Improved Faster R-CNN. In *2017 SAR in Big Data Era: Models, Methods and Applications (BIGSARDATA)*. 1–6. https://doi.org/10.1109/BIGSARDATA.2017.8124934

[74] Ke Li, Gang Wan, Gong Cheng, Liqiu Meng, and Junwei Han. 2020. Object Detection in Optical Remote Sensing Images: A Survey and a New Benchmark. *ISPRS Journal of Photogrammetry and Remote Sensing* 159 (Jan. 2020), 296–307. https://doi.org/10.1016/j.isprsjprs.2019.11.023

[75] Lin Li, Shengbing Zhang, and Juan Wu. 2019. Efficient Object Detection Framework and Hardware Architecture for Remote Sensing Images. *REMOTE SENSING* 11, 2376 (Oct. 2019). https://doi.org/10.3390/rs11202376

[76] S Li, KY Sun, YK Luo, N Yadav, and K Choi. 2020. Novel CNN-Based AP2D-Net Accelerator: An Area and Power Efficient Solution for Real-Time Applications on Mobile FPGA. *ELECTRONICS* 9, 5 (May 2020). https://doi.org/10.3390/electronics9050832

[77] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C. Berg. 2016. SSD: Single Shot MultiBox Detector. In *Computer Vision – ECCV 2016*, Bastian Leibe, Jiri Matas, Nicu Sebe, and Max Welling (Eds.). Springer International Publishing, Cham, 21–37. https://doi.org/10.1007/978-3-319-46448-0_2

[78] Zhuang Liu, Hanzi Mao, Chao-Yuan Wu, Christoph Feichtenhofer, Trevor Darrell, and Saining Xie. 2022. A ConvNet for the 2020s. https://doi.org/10.48550/arXiv.2201.03545 arXiv:2201.03545

[79] S. Lopez, T. Vladimirova, C. Gonzalez, J. Resano, D. Mozos, and A. Plaza. 2013. The Promise of Reconfigurable Computing for Hyperspectral Imaging Onboard Systems: A Review and Trends. *Proc. IEEE* 101, 3 (2013), 698–722. https://doi.org/10.1109/JPROC.2012.2231391

[80] Lucas Amilton Martins, Felipe Viel, Laio Oriel Seman, Eduardo Augusto Bezerra, and Cesar Albenes Zeferino. 2024. A Real-Time SVM-based Hardware Accelerator for Hyperspectral Images Classification in FPGA. *MICROPROCESSORS AND MICROSYSTEMS* 104, 104998 (Feb. 2024). https://doi.org/10.1016/j.micpro.2023.104998

[81] Gonzalo Mateo-Garcia, Joshua Veitch-Michaelis, Lewis Smith, Silviu Vlad Oprea, Guy Schumann, Yarin Gal, Atılım Güneş Baydin, and Dietmar Backes. 2021. Towards Global Flood Mapping Onboard Low Cost Satellites with Machine Learning. *Scientific Reports* 11, 1 (March 2021), 7249. https://doi.org/10.1038/s41598-021-86650-z

[82] JP Matos-Carvalho, F Moutinho, AB Salvado, T Carrasqueira, R Campos-Rebelo, D Pedro, LM Campos, JM Fonseca, and A Mora. 2019. Static and Dynamic Algorithms for Terrain Classification in UAV Aerial Imagery. *REMOTE SENSING* 11, 21 (Nov. 2019). https://doi.org/10.3390/rs11212501

[83] Sorour Mohajerani. 2024. SorourMo/38-Cloud-A-Cloud-Segmentation-Dataset.

[84] T Myojin, S Hashimoto, and N Ishihama. 2020. Detecting Uncertain BNN Outputs on FPGA Using Monte Carlo Dropout Sampling. In *Japan Aerospace Exploration Agency (JAXA)*, I Farkas, P Masulli, and S Wermter (Eds.), Vol. 12397. 27–38. https://doi.org/10.1007/978-3-030-61616-8_3

[85] Markus Nagel, Marios Fournarakis, Rana Ali Amjad, Yelysei Bondarenko, Mart van Baalen, and Tijmen Blankevoort. 2021. A White Paper on Neural Network Quantization. https://doi.org/10.48550/arXiv.2106.08295 arXiv:2106.08295 [cs]

[86] Romén Neris, Adrián Rodríguez, Raúl Guerra, Sebastián López, and Roberto Sarmiento. 2022. FPGA-Based Implementation of a CNN Architecture for the On-Board Processing of Very High-Resolution Remote Sensing Images. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing* 15 (2022), 3740–3750. https://doi.org/10.1109/JSTARS.2022.3169330

[87] DD Nguyen, DT Nguyen, MT Le, and QC Nguyen. 2024. FPGA-SoC Implementation of YOLOv4 for Flying-Object Detection. *JOURNAL OF REAL-TIME IMAGE PROCESSING* 21, 3 (May 2024). https://doi.org/10.1007/s11554-024-01440-w

[88] Ty Nguyen, Ian D. Miller, Avi Cohen, Dinesh Thakur, Arjun Guru, Shashank Prasad, Camillo J. Taylor, Pratik Chaudhari, and Vijay Kumar. 2021. PennSyn2Real: Training Object Recognition Models Without Human Labeling. *IEEE Robotics and Automation Letters* 6, 3 (July 2021), 5032–5039. https://doi.org/10.1109/LRA.2021.3070249

[89] Shuo Ni, Xin Wei, Ning Zhang, and He Chen. 2023. Algorithm–Hardware Co-Optimization and Deployment Method for Field-Programmable Gate-Array-Based Convolutional Neural Network Remote Sensing Image Processing. *Remote Sensing* 15, 24 (Jan. 2023), 5784. https://doi.org/10.3390/rs15245784

[90] NVIDIA. 2019. NVIDIA Jetson Nano Module. https://developer.nvidia.com/embedded/jetson-nano.

[91] NVIDIA. 2025. *NVIDIA RTX BLACKWELL GPU Architecture V1.01*. Technical Report. https://images.nvidia.com/aem-dam/Solutions/geforce/blackwell/nvidia-rtx-blackwell-gpu-architecture.pdf

[92] Lucas Prado Osco, José Marcato Junior, Ana Paula Marques Ramos, Lúcio André de Castro Jorge, Sarah Narges Fatholahi, Jonathan de Andrade Silva, Edson Takashi Matsubara, Hemerson Pistori, Wesley Nunes Gonçalves, and Jonathan Li. 2021. A Review on Deep Learning in UAV Remote Sensing. *International Journal of Applied Earth Observation and Geoinformation* 102 (Oct. 2021), 102456. https://doi.org/10.1016/j.jag.2021.102456

[93] Matthew J. Page, David Moher, Patrick M. Bossuyt, Isabelle Boutron, Tammy C. Hoffmann, Cynthia D. Mulrow, Larissa Shamseer, Jennifer M. Tetzlaff, Elie A. Akl, Sue E. Brennan, Roger Chou, Julie Glanville, Jeremy M. Grimshaw, Asbjørn Hróbjartsson, Manoj M. Lalu, Tianjing Li, Elizabeth W. Loder, Evan Mayo-Wilson, Steve McDonald, Luke A. McGuinness, Lesley A. Stewart, James Thomas, Andrea C. Tricco, Vivian A. Welch, Penny Whiting, and Joanne E. McKenzie. 2021. PRISMA 2020 Explanation and Elaboration: Updated Guidance and Exemplars for Reporting Systematic Reviews. *BMJ* 372 (March 2021), n160. https://doi.org/10.1136/bmj.n160

[94] Adam Paszke, Abhishek Chaurasia, Sangpil Kim, and Eugenio Culurciello. 2016. ENet: A Deep Neural Network Architecture for Real-Time Semantic Segmentation. https://doi.org/10.48550/arXiv.1606.02147 arXiv:1606.02147

[95] R Pitonak, J Mucha, L Dobis, M Javorka, and M Marusin. 2022. CloudSatNet-1: FPGA-Based Hardware-Accelerated Quantized CNN for Satellite On-Board Cloud Coverage Classification. *REMOTE SENSING* 14, 13 (July 2022). https://doi.org/10.3390/rs14133180

[96] J. R. Quinlan. 1986. Induction of Decision Trees. *Machine Learning* 1, 1 (March 1986), 81–106. https://doi.org/10.1007/BF00116251

[97] Amir Raoofy, Gabriel Dax, Max Ghiglione, Martin Langer, Carsten Trinitis, Martin Werner, and Martin Schulz. 2021. Benchmarking Machine Learning Inference in FPGA-based Accelerated Space Applications. *Proceedings of the Workshop on Benchmarking Machine Learning Workloads co-located with IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS)* (2021).

[98] Emilio Rapuano, Gabriele Meoni, Tommaso Pacini, Gianmarco Dinelli, Gianluca Furano, Gianluca Giuffrida, and Luca Fanucci. 2021. An FPGA-based Hardware Accelerator for Cnns Inference on Board Satellites: Benchmarking with Myriad 2-Based Solution for the CloudScout Case Study. *REMOTE SENSING* 13, 1518 (April 2021). https://doi.org/10.3390/rs13081518

[99] R Ratnakumar and SJ Nanda. 2021. A High Speed Roller Dung Beetles Clustering Algorithm and Its Architecture for Real-Time Image Segmentation. *APPLIED INTELLIGENCE* 51, 7 (July 2021), 4682–4713. https://doi.org/10.1007/s10489-020-02067-7

[100] Joseph Redmon and Ali Farhadi. 2017. YOLO9000: Better, Faster, Stronger. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 7263–7271.

[101] Joseph Redmon and Ali Farhadi. 2018. YOLOv3: An Incremental Improvement. https://doi.org/10.48550/arXiv.1804.02767 arXiv:1804.02767 [cs]

[102] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. 2015. U-Net: Convolutional Networks for Biomedical Image Segmentation. In *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015*, Nassir Navab, Joachim Hornegger, William M. Wells, and Alejandro F. Frangi (Eds.). Springer International Publishing, Cham, 234–241. https://doi.org/10.1007/978-3-319-24574-4_28

[103] Vít Růžička, Anna Vaughan, Daniele De Martini, James Fulton, Valentina Salvatelli, Chris Bridges, Gonzalo Mateo-Garcia, and Valentina Zantedeschi. 2022. RaVÆn: Unsupervised Change Detection of Extreme Events Using ML on-Board Satellites. *Scientific Reports* 12, 1 (Oct. 2022), 16939. https://doi.org/10.1038/s41598-022-19437-5

[104] Sebastian Sabogal and Alan George. 2021. A Methodology for Evaluating and Analyzing FPGA-accelerated, Deep-Learning Applications for Onboard Space Processing. In *2021 Ieee Space Computing Conference (Scc)*. 143–154. https://doi.org/10.1109/SCC49971.2021.00022

[105] Shaibal Saha and Lanyu Xu. 2025. Vision Transformers on the Edge: A Comprehensive Survey of Model Compression and Acceleration Strategies. https://doi.org/10.48550/arXiv.2503.02891 arXiv:2503.02891 [cs]

[106] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. 2018. MobileNetV2: Inverted Residuals and Linear Bottlenecks. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 4510–4520. https://doi.org/10.1109/CVPR.2018.00474

[107] SDMS. 2010. MSTAR Dataset. https://www.sdms.afrl.af.mil/index.php?collection=mstar.

[108] Ahmad Shawahna, Sadiq M. Sait, and Aiman El-Maleh. 2019. FPGA-Based Accelerators of Deep Learning Networks for Learning and Classification: A Review. *IEEE Access* 7 (2019), 7823–7859. https://doi.org/10.1109/ACCESS.2018.2890150

[109] Sherin C. Shibi and R. Gayathri. 2021. Onboard Target Detection in Hyperspectral Image Based on Deep Learning with FPGA Implementation. *MICROPROCESSORS AND MICROSYSTEMS* 85, 104313 (Sept. 2021). https://doi.org/10.1016/j.micpro.2021.104313

[110] Felix Siegle, Tanya Vladimirova, Jørgen Ilstad, and Omar Emam. 2015. Mitigation of Radiation Effects in SRAM-Based FPGAs for Space Applications. *ACM Comput. Surv.* 47, 2 (Jan. 2015), 37:1–37:34. https://doi.org/10.1145/2671181

[111] Laurent Sifre and Stéphane Mallat. 2014. Rigid-Motion Scattering for Texture Classification. https://doi.org/10.48550/arXiv.1403.1687 arXiv:1403.1687 [cs]

[112] Karen Simonyan and Andrew Zisserman. 2015. Very Deep Convolutional Networks for Large-Scale Image Recognition. https://doi.org/10.48550/arXiv.1409.1556 arXiv:1409.1556

[113] HS Suh, J Meng, T Nguyen, SK Venkataramanaiah, V Kumar, Y Cao, and JS Seo. 2021. Algorithm-Hardware Co-Optimization for Energy-Efficient Drone Detection on Resource-Constrained FPGA. In *Arizona State University*. 133–141. https://doi.org/10.1109/ICFPT52863.2021.9609840

[114] Vivienne Sze, Yu-Hsin Chen, Tien-Ju Yang, and Joel S. Emer. 2017. Efficient Processing of Deep Neural Networks: A Tutorial and Survey. *Proc. IEEE* 105, 12 (Dec. 2017), 2295–2329. https://doi.org/10.1109/JPROC.2017.2761740

[115] Daniela Szwarcman, Sujit Roy, Paolo Fraccaro, Þorsteinn Elí Gíslason, Benedikt Blumenstiel, Rinki Ghosal, Pedro Henrique de Oliveira, Joao Lucas de Sousa Almeida, Rocco Sedona, Yanghui Kang, Srija Chakraborty, Sizhe Wang, Carlos Gomes, Ankur Kumar, Myscon Truong, Denys Godwin, Hyunho Lee, Chia-Yu Hsu, Ata Akbari Asanjan, Besart Mujeci, Disha Shidham, Trevor Keenan, Paulo Arevalo, Wenwen Li, Hamed Alemohammad, Pontus Olofsson, Christopher Hain, Robert Kennedy, Bianca Zadrozny, David Bell, Gabriele Cavallaro, Campbell Watson, Manil Maskey, Rahul Ramachandran, and Juan Bernabe Moreno. 2025. Prithvi-EO-2.0: A Versatile Multi-Temporal Foundation Model for Earth Observation Applications. https://doi.org/10.48550/arXiv.2412.02732 arXiv:2412.02732 [cs]

[116] Mingxing Tan and Quoc V. Le. 2020. EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks. https://doi.org/10.48550/arXiv.1905.11946 arXiv:1905.11946 [cs]

[117] VAMF Torres, BRA Jaimes, ES Ribeiro, MT Braga, EH Shiguemori, HFC Velho, LCB Torres, and AP Braga. 2020. Combined Weightless Neural Network FPGA Architecture for Deforestation Surveillance and Visual Navigation of UAVs. *ENGINEERING APPLICATIONS OF ARTIFICIAL INTELLIGENCE* 87 (Jan. 2020). https://doi.org/10.1016/j.engappai.2019.08.021

[118] Yaman Umuroglu, Nicholas J. Fraser, Giulio Gambardella, Michaela Blott, Philip Leong, Magnus Jahre, and Kees Vissers. 2017. FINN: A Framework for Fast, Scalable Binarized Neural Network Inference. In *Proceedings of the 2017 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*. 65–74. https://doi.org/10.1145/3020078.3021744 arXiv:1612.07119 [cs]

[119] G Upadhyay, S Ghosal, S Kart, K Jain, SH Shantala, JT LalitKrushna, S Srividhya, and SA Balwantrao. 2024. Design and Implementation of CNN-based Custom Net Architecture with Improved Inference Time for Realtime Remote Sensing Application. In *2024 IEEE SPACE, AEROSPACE AND DEFENCE CONFERENCE, SPACE 2024*. 647–651. https://doi.org/10.1109/SPACE63117.2024.10667853

[120] UPV/EHU. 2021. Hyperspectral Remote Sensing Scenes - Grupo de Inteligencia Computacional (GIC). https://www.ehu.eus/ccwintco/index.php/Hyperspectral_Remote_Sensing_Scenes.

[121] P Vitolo, A Fasolino, R Liguori, L Di Benedetto, A Rubino, and GD Licciardo. 2024. Real-Time On-board Satellite Cloud Cover Detection Hardware Architecture Using Spaceborne Remote Sensing Imagery. In *University of Salerno*, MF Carlsohn (Ed.), Vol. 13000. https://doi.org/10.1117/12.3017554

[122] Erwei Wang, James J. Davis, Peter Y. K. Cheung, and George A. Constantinides. 2019. LUTNet: Rethinking Inference in FPGA Soft Logic. https://doi.org/10.48550/arXiv.1904.00938 arXiv:1904.00938 [cs]

[123] Erwei Wang, James J. Davis, Ruizhe Zhao, Ho-Cheung Ng, Xinyu Niu, Wayne Luk, Peter Y. K. Cheung, and George A. Constantinides. 2020. Deep Neural Network Approximation for Custom Hardware: Where We've Been, Where We're Going. *Comput. Surveys* 52, 2 (March 2020), 1–39. https://doi.org/10.1145/3309551

[124] EY Wang and DH Qiu. 2019. Acceleration and Implementation of Convolutional Neural Network Based on FPGA. In *Capital Normal University*. 321–325. https://doi.org/10.1109/iccsnt47585.2019.8962428

[125] Xiyue Wang, Xinsheng Wang, Zhiquan Zhou, and Yanhong Song. 2024 AUG 26 2024. Fast Detection and Obstacle Avoidance on UAVs Using Lightweight Convolutional Neural Network Based on the Fusion of Radar and Camera. *APPLIED INTELLIGENCE* (2024 AUG 26 2024). https://doi.org/10.1007/s10489-024-05768-5

[126] X Wei, WC Liu, L Chen, L Ma, H Chen, and Y Zhuang. 2019. FPGA-Based Hybrid-Type Implementation of Quantized Neural Networks for Remote Sensing Applications. *SENSORS* 19, 4 (Feb. 2019). https://doi.org/10.3390/s19040924

[127] WB Wu, CH Cao, Y Li, RX Zhang, and SL Dong. 2021. Design and Implementation of Remote Sensing Object Detection System Based on MPSoC. In *AOPC 2021: OPTICAL SENSING AND IMAGING TECHNOLOGY*, Y Jiang, Q Lv, D Liu, D Zhang, and B Xue (Eds.), Vol. 12065. https://doi.org/10.1117/12.2606142

[128] Gui-Song Xia, Xiang Bai, Jian Ding, Zhen Zhu, Serge Belongie, Jiebo Luo, Mihai Datcu, Marcello Pelillo, and Liangpei Zhang. 2018. DOTA: A Large-Scale Dataset for Object Detection in Aerial Images. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 3974–3983.

[129] Xiaowei Xu, Xinyi Zhang, Bei Yu, Xiaobo Sharon Hu, Christopher Rowen, Jingtong Hu, and Yiyu Shi. 2021. DAC-SDC Low Power Object Detection Challenge for UAV Applications. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 43, 2 (Feb. 2021), 392–403. https://doi.org/10.1109/TPAMI.2019.2932429

[130] Réda Yahiaoui, Farid Alilat, and Saliha Loumi. 2017. Parallelization of Fuzzy ARTMAP Architecture on FPGA: Multispectral Classification of ALSAT-2A Images. *IEEE Transactions on Industrial Electronics* 64, 12 (Dec. 2017), 9487–9495. https://doi.org/10.1109/TIE.2017.2708028

[131] Tianwei Yan, Ning Zhang, Jie Li, Wenchao Liu, and He Chen. 2022. Automatic Deployment of Convolutional Neural Networks on FPGA for Spaceborne Remote Sensing Application. *Remote Sensing* 14, 13 (Jan. 2022), 3130. https://doi.org/10.3390/rs14133130

[132] G Yang, J Lei, WY Xie, ZM Fang, YS Li, JX Wang, and X Zhang. 2022. Algorithm/Hardware Codesign for Real-Time On-Satellite CNN-Based Ship Detection in SAR Imagery. *IEEE TRANSACTIONS ON GEOSCIENCE AND REMOTE SENSING* 60 (2022). https://doi.org/10.1109/TGRS.2022.3161499

[133] RH Yang, ZK Chen, BA Wang, YF Guo, and LT Hu. 2023. A Lightweight Detection Method for Remote Sensing Images and Its Energy-Efficient Accelerator on Edge Devices. *SENSORS* 23, 14 (July 2023). https://doi.org/10.3390/s23146497

[134] Yi Yang and Shawn Newsam. 2010. Bag-of-Visual-Words and Spatial Extensions for Land-Use Classification. In *Proceedings of the 18th SIGSPATIAL International Conference on Advances in Geographic Information Systems (GIS '10)*. Association for Computing Machinery, New York, NY, USA, 270–279. https://doi.org/10.1145/1869790.1869829

[135] Zhewei Yao, Zhen Dong, Zhangcheng Zheng, Amir Gholami, Jiali Yu, Eric Tan, Leyuan Wang, Qijing Huang, Yida Wang, Michael Mahoney, and Kurt Keutzer. 2021. HAWQ-V3: Dyadic Neural Network Quantization. In *Proceedings of the 38th International Conference on Machine Learning*. PMLR, 11875–11886.

[136] Q Yu, A Liu, XX Yang, and WM Diao. 2024. An Improved Lightweight Deep Learning Model and Implementation for Track Fastener Defect Detection with Unmanned Aerial Vehicles. *ELECTRONICS* 13, 9 (May 2024). https://doi.org/10.3390/electronics13091781

[137] Bingyi Zhang, Rajgopal Kannan, Viktor Prasanna, and Carl Busart. 2022. Accurate, Low-latency, Efficient SAR Automatic Target Recognition on FPGA. In *2022 32nd International Conference on Field-Programmable Logic and Applications (FPL)*. IEEE Computer Society, 1–8. https://doi.org/10.1109/FPL57034.2022.00013

[138] BY Zhang, R Kannan, V Prasanna, and C Busart. 2023. Accelerating GNN-based SAR Automatic Target Recognition on HBM-enabled FPGA. In *University of Southern California*. https://doi.org/10.1109/HPEC58863.2023.10363615

[139] Ning Zhang, Shuo Ni, Tingting Qiao, Wenchao Liu, and He Chen. 2023. An Extremely Pipelined FPGA-based Accelerator of All Adder Neural Networks for on-Board Remote Sensing Scene Classification. In *2023 International Conference on Field Programmable Technology, Icfpt (IEEE International Conference on Field Programmable Technology)*. 258–261. https://doi.org/10.1109/ICFPT59805.2023.00036

[140] Ning Zhang, Xin Wei, He Chen, and Wenchao Liu. 2021. FPGA Implementation for CNN-based Optical Remote Sensing Object Detection. *ELECTRONICS* 10, 282 (Feb. 2021). https://doi.org/10.3390/electronics10030282

[141] XL Zhang, X Wei, QB Sang, H Chen, and YZ Xie. 2020. An Efficient FPGA-Based Implementation for Quantized Remote Sensing Image Scene Classification Network. *ELECTRONICS* 9, 9 (Sept. 2020). https://doi.org/10.3390/electronics9091344

[142] Qiang Zhao, Le Yu, Zhenrong Du, Dailiang Peng, Pengyu Hao, Yongguang Zhang, and Peng Gong. 2022. An Overview of the Applications of Earth Observation Satellite Data: Impacts and Future Trends. *Remote Sensing* 14, 8 (Jan. 2022), 1863. https://doi.org/10.3390/rs14081863

[143] Yonghui Zhao, Yong Lv, and Chao Li. 2023. Hardware Acceleration of Satellite Remote Sensing Image Object Detection Based on Channel Pruning. *APPLIED SCIENCES-BASEL* 13, 10111 (Sept. 2023). https://doi.org/10.3390/app131810111

[144] Daquan Zhou, Qibin Hou, Yunpeng Chen, Jiashi Feng, and Shuicheng Yan. 2020. Rethinking Bottleneck Structure for Efficient Mobile Network Design. In *Computer Vision – ECCV 2020*, Andrea Vedaldi, Horst Bischof, Thomas Brox, and Jan-Michael Frahm (Eds.). Springer International Publishing, Cham, 680–697. https://doi.org/10.1007/978-3-030-58580-8_40

[145] Xiao Xiang Zhu, Devis Tuia, Lichao Mou, Gui-Song Xia, Liangpei Zhang, Feng Xu, and Friedrich Fraundorfer. 2017. Deep Learning in Remote Sensing: A Comprehensive Review and List of Resources. *IEEE Geoscience and Remote Sensing Magazine* 5, 4 (Dec. 2017), 8–36. https://doi.org/10.1109/MGRS.2017.2762307

Received 20 February 2007; revised 12 March 2009; accepted 5 June 2009