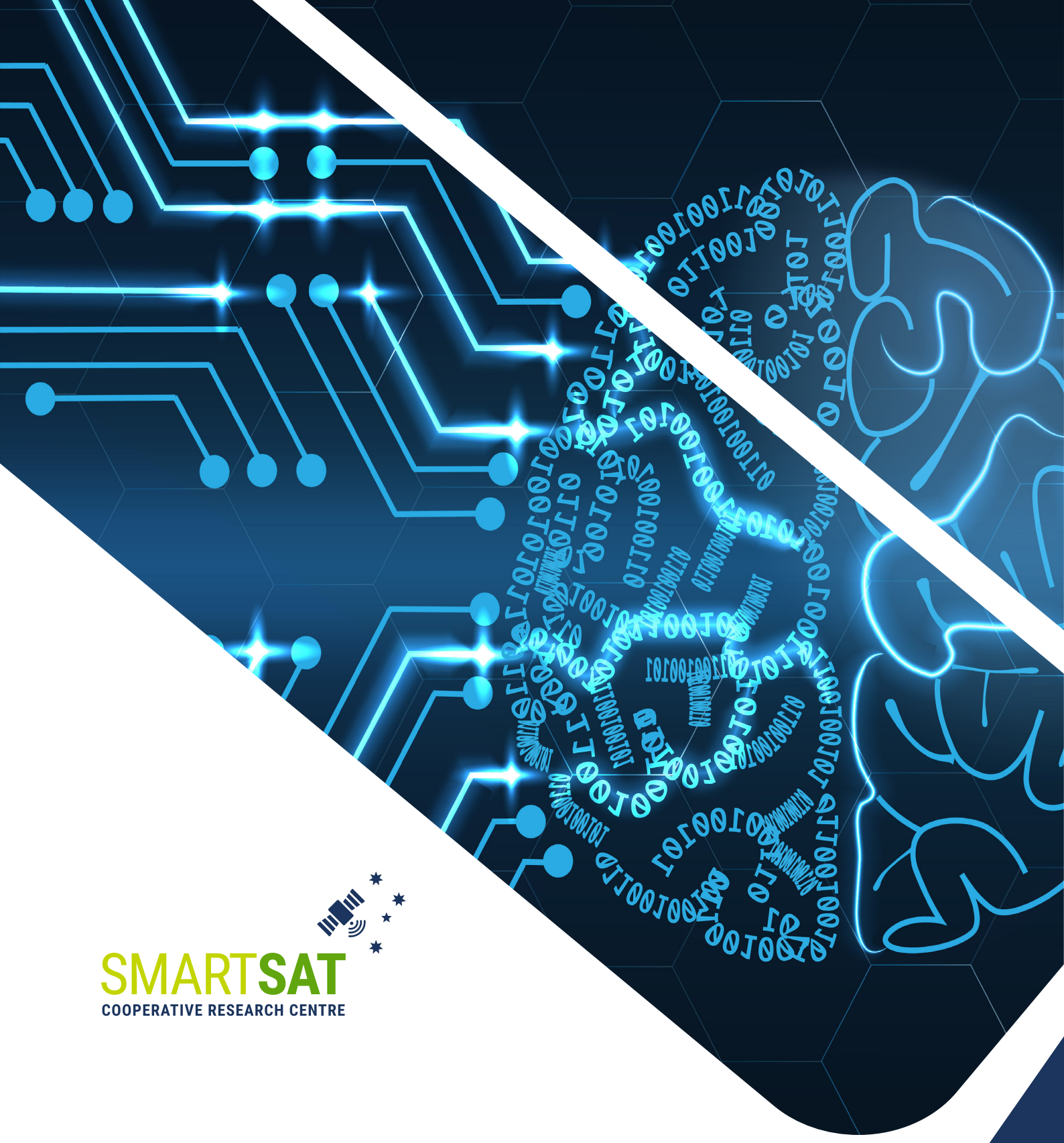

SMARTSAT
COOPERATIVE RESEARCH CENTRE

TECHNICAL REPORT 06

Machine Learning Onboard Satellites

Technical Report No. 10

Machine Learning Onboard Satellites

November 2021

ISBN:

This report should be cited as:

SmartSat 2021, Machine Learning Onboard Satellites, SmartSat Technical Report no. 010, SmartSat, Adelaide, Australia.

Disclaimer:

This publication is provided for the purpose of disseminating information relating to scientific and technical matters. Participating organisations of SmartSat do not accept liability for any loss and/or damage, including financial loss, resulting from the reliance upon any information, advice or recommendations contained in this publication. The contents of this publication should not necessarily be taken to represent the views of the participating organisations.

Acknowledgement:

SmartSat acknowledges the contribution made by Aninda Saha and Alina Bialkowski (The University of Queensland), Yu Sun and Kai Qin (Swinburne University of Technology), and Kien Nguyen and Clinton Fookes (Queensland University of Technology) towards the writing and compilation of this technical report.

Executive Summary

Over the last decade, machine learning (ML) has been a key driver of many data-driven applications. As such, the rapidly growing space industry is poised to take advantage of recent ML advancements to automate much of its data processing. This includes satellite-based applications such as earth observation, communications, navigation as well as autonomous failure detection and recovery of spacecraft. Key ML algorithms such as object detection, semantic segmentation, pose estimation and anomaly detection help enable these space applications. However, many of these algorithms i.e., the trained models, impose large computational workloads requiring large, power-hungry GPUs to execute, which is at odds with operating in a space environment. On the other hand, downlinking data for processing on earth is also not an option for many satellite applications that require low-latency solutions. Edge computing is the efficient processing solution at the source of the data, which could be the key to enabling widespread adoption of ML for satellite applications. Moreover, by reducing the need to offload sensitive data, onboard processing can alleviate privacy-related barriers to ML adoption in space.

While promising, onboard processing presents its own set of challenges primarily due to its low computing resources imposed by the size, weight, volume and power constraints of satellite platforms. Therefore, to deploy high-quality ML models to computing devices onboard satellite platforms, they must be designed for efficiency without compromising accuracy, which often arises due to the accuracy-speed trade-off phenomenon in ML literature. However, it has been discovered that most state-of-the-art models are quite “wasteful” in that they often do not make optimal use of their parameter space. This provides an opportunity to either apply model compression techniques to remove redundant parameters from larger models, or design compact models with high accuracy from the outset using Neural Architecture Search (NAS). Hardware acceleration is another technique that aims to speed up computations at a hardware level, either by parallelising data processing digitally or by employing analog computing techniques to physically speed up the signal propagation. This report discusses the details of different compression and acceleration techniques and how they can be codesigned to increase efficiency for space applications.

With a growing demand for edge computing, there has been a surge in the availability of off-the-shelf hardware platforms and frameworks that support model compression and hardware acceleration. This report discusses such platforms and frameworks in detail and outlines the pros and cons of the different options. The key metrics that determine the choice of hardware platforms for a given application are the floating-point operations (FLOPs), memory requirements and performance per watt of the model. The choice of ML development framework follows from the choice of hardware, which determines the operating system capable of being flashed onto the hardware.

Additionally, one of the biggest technical barriers to ML deployment in space is that space is filled with extreme radiation and temperature which can interfere with flight computers. Extreme radiation can cause bit flips, effectively corrupting computations. Radiation hardening and other shielding techniques are often necessary to get around such challenges. However, shielding can be quite expensive and add significant size and weight to volume-constrained satellite platforms. Therefore, radiation-tolerant designs using off-the-shelf computers may often be preferred due to their lower cost, smaller form factors and greater software support over radiation-hardened hardware. Examples of radiation-tolerant designs include redundant computing to perform self-checking to counter the effects of radiation.

In summary, with a rapidly growing space industry, ML has a huge role to play in automating the processing of the exorbitant amount of data collected from space every day. Although the adoption of ML algorithms for space applications have lagged, significant strides are now being taken by key organisations to encourage the deployment of ML to space environments. Therefore, it is envisaged that the power of ML can be leveraged for space applications to deliver value to society.

Table of Contents

1. Introduction

Since the launch of the first artificial satellite, Sputnik 1, by the Soviet Union in 1957, satellites have found many applications that have benefited large segments of the world population. For example, Ariane 5 and GOES-16 satellites have been used for telecommunications and weather forecasting respectively, both of which are technologies that modern life heavily depends on. However, these traditional satellites have been limited in their scalability due to their exorbitant costs. With recent miniaturisation of computers and other hardware, hundreds of satellites are being launched into space every year at a fraction of the cost. This has significantly reduced the barrier to entry into space for both commercial and military applications. Satellites offer a unique vantage point with unobstructed lines of sight and communication for various remote sensing tasks as well as reconnaissance missions. However, it is often impractical to perform manual analysis on the large volumes of data that is captured. Therefore, an automated data processing solution is required to offload this analysis task to computers. Machine learning (ML) has been grabbing headlines recently due to its ever-improving pattern recognition capability enabled by deep learning (DL). This has opened the scope of automation by enabling computers to learn from data and make more intelligent decisions. As such, deep learning-based computer vision and reinforcement learning techniques are starting to be used to automate data processing in space applications.

Certain space applications have strict requirements of needing to be real-time, consuming minimal energy and not disseminating sensitive data through openly accessible communication channels. The current strategy of downlinking data to Earth for processing violates all three requirements. This calls for *onboard processing* techniques, which involve executing data processing algorithms on a computing chip integrated onto a target platform. A related concept called *edge computing* refers to performing computations close to the source of the data, typically using mobile or IoT devices. The term *on-orbit processing* will be used to refer to edge processing onboard satellites in a space environment. By eliminating the need to downlink data to Earth, on-orbit processing can reduce latency and power consumption and provide increased data security. However, rapid satellite miniaturisation has meant that deployable computing platforms are now even more volume and power constrained, which in turn results in lower computing resources. With the high computational workload of DL algorithms, it becomes a significant challenge to deploy these models to space using on-orbit computing platforms.

On-orbit ML is a relatively new initiative, with PhiSat-1 being the first ever satellite launched with such capabilities, in 2020. As such, there are only a handful of survey papers that attempt to summarise the applications of ML in space. Hoeser et al. [1] published a two-part survey of the algorithms and applications of Earth Observation (EO). Meanwhile, Fourati et al. [2] recently published a review paper discussing the applications of ML for satellite communications. While great pieces of work, both of these papers neglect to address the challenges surrounding on-orbit processing using deployable flight computers in space. Similarly, other relevant works either address space applications without addressing the challenges of on-orbit computing [3], [4] or vice-versa [5], [6]. Therefore, this report addresses this gap by conducting a full-suite review that discusses: (1) space applications that require on-orbit processing; (2) algorithmic complexities and the available computing platforms capable of meeting such demands; (3) software toolchains for running ML inferences on edge devices; (4) model compression and hardware acceleration techniques; and (5) mitigation strategies for the challenges posed to computing hardware by the space environment. Table 1 presents a high-level summary of this report's contributions and provides a comparison with the existing literature.

Table 1: Summary of survey contribution

Source	Space Applications	Algorithms	Edge Computing	Hardware Platforms
Object Detection and Image Segmentation with Deep Learning on Earth Observation Data: A Review-Part I: Evolution and Recent Trends [1]	✓	✓	x	x
Artificial intelligence for satellite communication: A review [2]	✓	✓	x	x
Computer vision algorithms and hardware implementations: A Survey [5]	x	✓	✓	✓
Machine Learning at the Network Edge: A Survey [6]	x	x	✓	✓
The Final Frontier: Deep Learning in Space [3]	✓	x	✓	x
Accelerating Deep Learning Applications in Space [4]	✓	✓	x	✓
This report	✓	✓	✓	✓

The remainder of this report will cover the topics presented in Table 1 in further detail. Section II discusses in detail a few key space applications and how the benefits of on-orbit computing, such as increased autonomy, increased data security, reduced latency and reduced power consumption, can play a key role in such applications. Next, Section III discusses the details of the relevant algorithms that enable the key space applications, their computational and memory requirements and how they can be made edge-deployable using model compression and hardware acceleration techniques. This section also discusses the available edge-ML frameworks that provide compression and acceleration support. Section IV discusses the available hardware platforms able to support computationally expensive ML models, their ability to operate in the harsh space environment and the mitigation strategies from extreme radiation and temperature. Finally, Section V provides a future outlook outlining the challenges, opportunities and trends in adopting space ML to deliver meaningful business value.

This review considers journal and conference papers covering a date range from 2011 to 2021, to capture the developments over the most recent ML boom on the following topics:

1. Space ML applications that require edge processing;
2. Algorithms and frameworks required for deployment on the edge;
3. Hardware suitable for edge ML in space environments; and
4. Algorithmic and hardware-related challenges of operating in space environments and ways to mitigate them.

2. Onboard ML in Space: An Overview

2.1 Motivation

Despite the rapid developments in ML, space applications have not seen the same pace of uptake for these technologies. Part of this has been driven by a lack of commercial space activity due to the excessively high cost of satellite launches. However, due to hardware miniaturisation and a rapidly growing commercial space industry, instigated by companies like SpaceX and Northrop Grumman, the cost of satellite launches has been reduced dramatically. This cost reduction has enabled the launch of CubeSats by commercial businesses and academic institutions alike [1]. CubeSats are a special class of camera-equipped miniaturised satellites that are commonly used as teaching tools and early technology demonstrations [7]. These satellites are widely used due to their 10,000x cost reduction factor compared to full-scale satellites [8]. With a dimension of only 10×10×10 cm, these satellites place a significant constraint on the power consumption and form factor of its internal electronics [2]. Considering these limitations, an investigation of compact, low-resourced, energy-efficient hardware and software techniques capable of executing object detection in real time is required to enable intelligent image processing onboard CubeSats.

It was not until 2020 that the European Space Agency (ESA) pioneered the deployment of ML into space for earth observation onboard a 6U CubeSat called Φ-Sat-1 using an Intel Movidius chip. This particular mission was conducted to demonstrate how on-orbit ML could be applied to improve the data transmission process [3]. They achieved this by deploying a real-time cloud detection model capable of filtering out images that are too heavily occluded by clouds to provide any useful information about the terrain. In doing so, through pre-processing and filtering noisy data onboard the satellite, significant power and bandwidth savings were achieved by down-linking only useful data to Earth.

With Φ-Sat-1 showing tremendous promise, companies around the world are gearing up for the next phase of ML deployments to space. As a testament to the rapidly growing interest in on-orbit ML, IBM has recently partnered with NASA to develop a custom on-board computing solution in space [4]. This development can avail several opportunities including increased autonomy, reduced latency, reduced power consumption and improved data security. These benefits over the traditional method of down-linking data for processing on large servers on the ground are illustrated in Figure 1. Increased autonomy is crucial for satellites needing to react in real time to perform collision avoidance manoeuvres to evade the growing amounts of space debris. On the other hand, reduced power consumption is also critical due to the tight power budget onboard satellites. Finally, data security is paramount when handling with sensitive satellite data, which could be compromising to governments, companies or individuals if leaked.

[1] https://www.nasa.gov/sites/default/files/atoms/files/nasa_csli_cubesat_101_508.pdf
[2] https://www.nasa.gov/content/what-are-smallsats-and-cubesats
[3] https://directory.eoportal.org/web/eoportal/satellite-missions/p/phisat-1
[4] https://www.ibm.com/cloud/blog/ibm-develops-a-unique-custom-edge-computing-solution-in-space

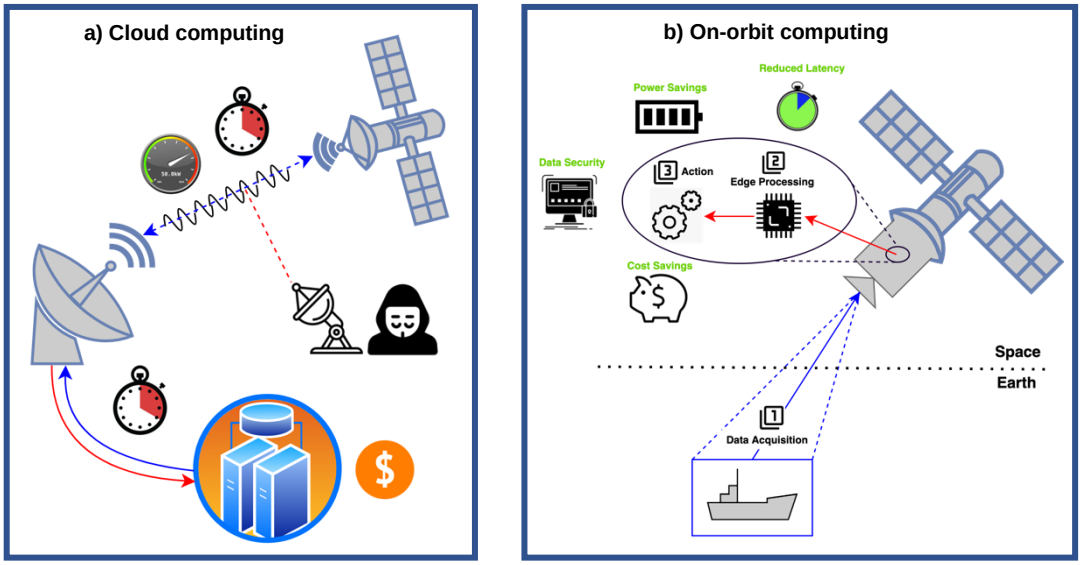

Figure 1: Motivation for on-orbit computing. The left image (a) indicates the drawbacks of over-reliance on cloud computing for processing satellite workloads. In addition to incurring latency and high power draw, it remains susceptible to interceptions by hackers. On the other hand, the right-hand image (b) indicates that on-orbit computing can overcome these challenges by processing at the edge without off-loading sensitive or noisy data.

2.2 Applications

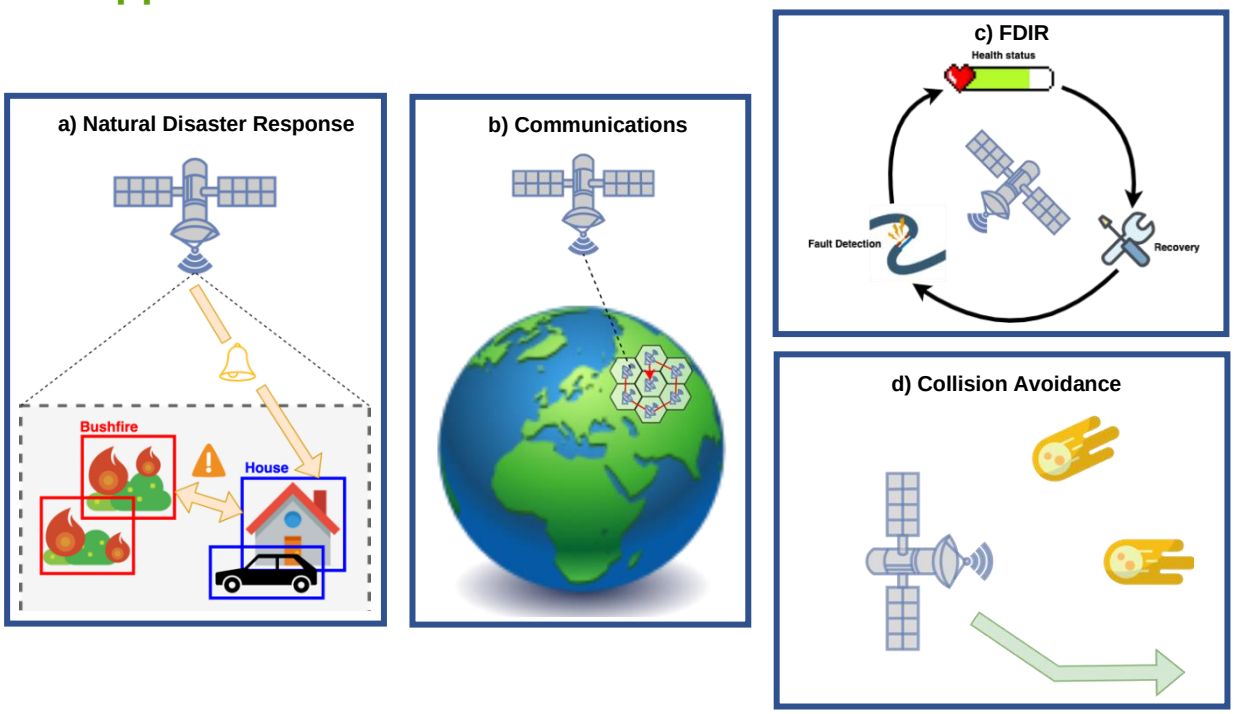

Figure 2: Applications of machine learning in space. Image (a) depicts the use of ML for real-time natural disaster response using satellite imagery. Image (b) demonstrates the use of ML for intelligently managing satellite communication demands using beam hopping techniques. Image (c) illustrates the use of continuous diagnosis of satellite health and autonomous recovery capability. Image (d) highlights the use of real-time ML for collision and obstacle avoidance tasks.

Several space applications could benefit from on-orbit processing including earth observation for natural disaster response, communications, Fault Detection, Isolation and Recovery (FDIR) as well as collision avoidance. These applications are illustrated in Figure 2 and are detailed in the following paragraphs.

With companies such as Planet [5] delivering high-quality Earth Observation (EO) data, companies and governmental agencies can now cost-effectively conduct important tasks such as flood detection, vegetation monitoring, settlement tracking, traffic monitoring and many more. Information gathered from these tasks have informed various commercially beneficial applications such as urban planning, natural disaster response, tracking climate change patterns, understanding customer behaviour, and assisting with law enforcement. Particularly, natural disaster response is an area of untapped potential due to the current lack of real-time information, which can be derived from satellite imagery. Satellite imagery provides the unique advantage of aerial vision which can be exploited for efficient deployment of resources for disaster management. Such benefits make EO a key motivation for many organisations.

Satellite communications is another application that is garnering significant attention, thanks to companies like Starlink, OneWeb, Telesat and Project Kuiper, with their vision to provide global internet coverage with satellites. One of the biggest challenges of terrestrial communication networks is that it is very expensive to reach the remote regions of the world. To achieve this, Starlink have deployed a constellation of interconnected satellites to lower earth orbit (LEO) to provide low-latency communications with global coverage. As low-latency is a key requirement of communication systems, on-orbit processing is critical for these applications.

The European Space Agency (ESA) has also developed capabilities for performing Fault Detection, Isolation and Recovery (FDIR) [9] which involves continuous monitoring of flight performance, anomaly detection in spacecraft performance and autonomous failure management. Real-time asset health monitoring is critical for early mitigation of electronic and mechanical failures, which if left unchecked could result in mission failure. Therefore, on-orbit processing can play a key role in enabling real-time FDIR.

Moreover, with a growing network of satellites in orbit as well as increasing amounts of space debris, collision avoidance manoeuvres are becoming increasingly routine. However, these tasks often require round-the-clock oversight from expert teams, which can be quite costly. Therefore, ESA is currently encouraging the global ML community to develop autonomous collision avoidance systems. Exploiting the strong predictive capabilities of neural networks, early results showed promise in replicating the decision process and correctly predicting when a collision avoidance manoeuvre was necessary [6]. As the space environment is highly dynamic, the increased autonomy afforded by on-orbit processing can enable satellites to safely navigate themselves, in the event of communication dropouts with ground stations.

Therefore, in light of the myriad benefits mentioned above, on-orbit processing is positioned to play a key role in future deployments of ML systems into space. This prompts the need for a comprehensive survey of on-orbit processing techniques in space environments. Due to the nascency of on-orbit ML, this report highlights the existing implementations as well as the opportunities that can be derived from future deployments.

[5] https://www.planet.com/

[6] https://www.space.com/AI-autonomous-space-debris-avoidance-esa

3. On-orbit ML for Space: Techniques

3.1 Space ML Techniques

Deploying intelligent systems to space environments provides several benefits. To understand how exactly these benefits are leveraged, one must first understand the core algorithms behind these applications and the challenges that come with them. Some of the key algorithms include image classification, object detection, semantic segmentation, pose estimation and anomaly detection among others. This subsection will step through how these algorithms are applied in space applications and some of the domain-specific challenges that render state-of-the-art models suitable or unsuitable for such applications.

3.1.1 Image Classification

Image classification largely aims to classify images into their object classes. Depending on the application in a space setting, the object categories can vary such as land cover [10], asteroid and cloud [11], among many others. Classification tasks on satellite imagery can inherit a wide range of existing architectures that have already been applied to ground-based natural images such as VGG [12], Inception [13], ResNet [14], EfficientNet [15] and NASNet [16], to name a few. The main challenge in classification in space is the lack of large-scale datasets. For instance, while large-scale natural datasets such as ImageNet contain 14 million images [17], the largest satellite imagery dataset only contains 0.6 million patches [7]. To deal with this challenge, techniques such as semi-supervised learning [18] and knowledge distillation [19] are useful. The gap between natural and satellite imagery can also be solved using transfer learning [20] and domain adaptation [21].

3.1.2 Object Detection

Object detection involves ML techniques locating and classifying an object based on a range of predefined categories. In the context of earth observation from space, objects to be located include large objects such as roads, vehicles, buildings, etc. Compared to object detection on ground-based natural images, satellite aerial imagery exhibits some unique challenges including: (1) a dearth of labelled training data, (2) low ground resolution, (3) small spatial extent, (4) arbitrary rotation angles, (5) novel overhead top view, and (6) non-uniform object distribution [1]. Satellite imagery object detection datasets such as DOTA [22], xView [23] and SpaceNet [24] have also revealed additional challenges such as ultra-high image resolutions, extreme class imbalances and very sparse annotations. The challenges need to be considered when designing detectors. For example, two-stage detectors such as Faster RCNN [25] and Cascade RCNN [26] are more accurate than one-stage detectors. In addition, two-stage detectors are more robust to object scales, which is especially important to deal with small-resolution objects in the space setting. In contrast, one-stage detectors such as YOLO [27], [28] and SSD [29] are faster at the cost of lower accuracy than the two-stage counterparts.

The limited computing resources in an onboard setting demands detection models to be efficient, and often have to trade off against processing time, memory, number of computational operations and energy consumption. To ensure edge deployability, techniques such as model compression [30]–[33] and compact model design using Neural Architecture Search (NAS) [34], [35] are relevant. Convolutional Neural Networks (CNNs) usually contain tens to hundreds of millions of parameters with hundreds of layers, e.g., ResNet-101 has 44.6M parameters and 347 layers [14]. Designing lightweight CNNs is an important step in deploying resource-constrained computing platforms. Lightweight CNNs employ advanced techniques to efficiently trade off between resource and accuracy, minimising their model size and computations in term of the number of floating-point operations (FLOPs), while retaining high accuracies. Specialised lightweight CNN architectures for edge processing include MobileNets [36],

[7] http://bigearth.net/

ShuffleNet [37], and PeleeNet [38]. Another noteworthy architecture is DarkNet, which has been the key backbone behind the modern real-time object detection series called YOLO [27], [28].

3.1.3 Semantic Segmentation

Semantic segmentation reaches beyond a rectangular bounding box around an object as in detection by learning a class label for each pixel in an image. While segmentation in space also exhibits the same challenges as object detection, there are two additional challenges that significantly impact the segmentation performance: (1) the low quality of data due to such artifacts as motion blur, cloudy and out-of-focus images; and (2) the high density of objects in an image. There are two main approaches in semantic segmentation: (1) Deep Convolutional Networks (DCNs), and (2) Probabilistic Graphical Models. Most DCN-based approaches such as FCN [39] and U-Net [40] employ an encoder followed by a decoder to distil semantic knowledge for segmentation. Graphical Models such as CRF-RNN [41] and Deeplab-v3 [42] learn structured output of the semantic segmentation task via probabilistic theory. The high density of pixels to be classified could be challenging for the DCN-based approaches since the encoding process may discard small object details. To deal with the low-quality data challenge, modelling uncertainty is key. From this perspective, approaches such as Bayesian SegNet [43] will play a crucial role in space segmentation applications. Another key challenge for segmentation in space is the lack of large-scale datasets and the high cost of collection and annotation. Unsupervised or weakly-supervised segmentation approaches [44], [45] could be key in tackling these challenges.

3.1.4 Pose Estimation

A wide range of space missions involve close-proximity operations with uncooperative space objects such as satellites [46], [47], space debris (e.g., active debris removal), and comets and asteroids (space exploration) [48]. Interacting with these objects requires an estimation of their relative position and attitude, referred to as pose. Modern pose estimation networks such as TransPose [49] and OmniPose [50] can effectively infer the position and orientation of an object either in 2D or 3D space [51]. Compared to general pose estimation, pose estimation in space presents several unique challenges, such as: (1) the use of single monocular cameras; (2) objects varying in a wide range of scales; (3) noisy images; and (4) cluttered backgrounds. Due to these challenges, model-based approaches are preferred for pose estimation in space rather than appearance-based methods [46], [48]. Data augmentation and noise filtering are also important pre-processing steps to improve the overall performance. The development of large-scale pose estimation datasets such as [52] could significantly boost the research in this area.

3.1.5 Hyperspectral Tasks

One unique challenge in space ML is the presence of multispectral and hyperspectral data along with the popular optical RGB data [8]. Performing ML tasks such as detection, segmentation and classification using multispectral or hyperspectral data exhibit unique challenges compared to the RGB data including: (1) high dimensionality, and (2) nonlinear and complex data due to different atmospheric and geometric distortions [53]. High dimensionality is the key challenge. For example, the HyperScout2 sensor has 45 channels in the spectral range 400 – 1000 nm and 3 channels in the spectral range 8000 – 14000 nm [9]. While the high dimensionality has huge advantages in rich details and ability to see unseen details, they pose significant difficulties in making sense of the huge data, especially in removing redundancy, extracting useful data, and reducing noise [54].

[8] https://gisgeography.com/hyperspectral-imaging/
[9] https://www.cosine.nl/cases/hyperscout-2/

3.2 On-orbit ML Techniques

Although the training phase of ML models is typically performed on large servers with powerful GPUs, the application of those models requires on-orbit computing capabilities. On-orbit processing falls within the broader realm of edge computing, which presents a common set of benefits and challenges. For instance, processing data using edge devices generally lends itself to lower latency decision making as it decreases the need to downlink data to servers on the ground. Reducing the need for server communication also reduces power consumption and improves autonomy as systems can operate without having to rely on uninterrupted network connectivity. Moreover, edge computing devices typically tend to have small form factors, which is particularly beneficial for on-orbit processing due to the limited real estate on satellite platforms. Finally, edge computing tends to provide increased data security as it does not require all the data to be transmitted to a cloud server, reducing the risk of it being intercepted. Data security is a primary concern for satellite data as it can often capture sensitive information that must be protected.

However, edge computing comes with two main drawbacks of lower compute and lower memory resources, mainly due to the smaller form factor of edge-class devices. Therefore, ML models deployed to the edge necessarily need to be designed for efficiency, both in terms of compute as well as memory requirements. There are two main ways of doing this, either by compressing high-performing pre-trained models into more compact ones or by designing efficient architectures from scratch that are parameter and compute efficient, commensurate with the requirements for the particular application. An alternative way to optimise neural network throughput is to perform hardware acceleration by applying sophisticated software-hardware co-design techniques to exploit the model compression algorithms at a hardware level. Therefore, this section will discuss the different compression and architecture search techniques as well as the hardware acceleration techniques that enable practical speed enhancements.

Compression

Modern ML architectures are usually computationally expensive and large with hundreds of layers and millions of parameters. However, studies have found that most of these large models tend to be over-parameterised, leading to lots of redundant parameters and operations in the network. This has motivated a hot trend looking to remove these redundancies from the models, either to fit into resource-limited devices such as mobile phones and embedded systems or to target real-time applications, with a small performance trade-off [30].

Model compression can be achieved by parameter pruning and sharing, quantisation and binarisation, low-ranked factorisation and knowledge distillation [55]. Pruning and quantisation are two popular techniques that have been implemented in popular deployment frameworks such as TensorfowLite and TensorRT to enable on-device inference. Each method has its pros and cons in its ease of achieving memory savings and speed enhancements. Therefore, the choice of compression technique depends on whether the deployment platform is more memory or compute constrained.

Pruning involves the removal of less-significant weight connections from a neural network using some kind of scoring system, typically L1 norm, to assess their importance [31]. This is typically done at a weight, neuron, channel or layer level [56]. Pruning at a weight level is a form of unstructured pruning which leads to sparse weight matrices that does not easily translate to speed enhancement due to the lack of hardware or framework support for sparse matrix multiplication [57]. On the other hand, pruning at a neuron, channel or layer level typically leads to the deletion of whole rows, columns and matrices, which leads to both memory and compute savings. However, this comes at the expense of a coarser model approximation, leading to a larger accuracy trade-off.

Quantisation is the technique of mapping a set of higher precision set of weights to a lower precision set by rounding the former set to the closest value in the latter set [31]. This is performed to reduce computation, memory and power utilisation while aiming retain the maximum prediction accuracy. An

example would be quantising 32-bit floating point numbers to 8-bit fixed point numbers [58]. This would clearly provide a 4x memory saving as well as speed enhancements due to the reduced complexity of lower precision operations. While models can theoretically be quantised down to binary levels, similar to Binary Neural Networks (BNNs) [59], [60], the barrier to achieving speed enhancements is that lower precision operations must be supported by the target hardware and framework to achieve practical speedups. Fortunately, most of the commonly used operations in deep learning are supported by popular frameworks, such as Tensorflow, PyTorch and MXNet [61].

Low-rank factorisation reduces model complexity by attempting to replace large groups of redundant filters with a smaller group of eigen-filters, where the original filters can be approximated by some linear combination of the eigen-filters [32]. This can be achieved through variants of the Singular Value Decomposition (SVD) algorithm which is used to find the eigen-filters [62]. By reducing the number of filters, this method simultaneously reduces the parameter count as well as the number of expensive convolution operations. Therefore, unlike pruning and quantisation, this is a hardware and framework-agnostic model compression technique.

Knowledge distillation (KD) is a class of end-to-end compression techniques which reduce model complexity by training a compact student network to mimic a larger and higher-performance teacher network [33]. This technique was originally proposed in the context of image classification with the idea that the teacher encodes a more accurate representation, termed “dark knowledge”, of inter-class similarities than the one-hot encoded ground truth labels, which assigns a value of 1 to the correct class and 0 to all others. By distilling the dark knowledge, smaller and more memory efficient student networks can often achieve comparable accuracy to their teacher networks.

AutoML and Neural Architecture Search (NAS)

AutoML is a tool that aims to automate the entire pipeline of all machine learning techniques, not just deep learning [34]. The AutoML pipeline covers data preparation, feature engineering, hyperparameter optimisation and architecture optimisation, where the architecture optimisation of neural networks is termed Neural Architecture Search (NAS) [34]. NAS automates the process of architecture design of neural networks by iteratively sampling a population of child networks, evaluating the child models’ performance metrics as rewards and learning to generate high-performance architecture candidates [63]. NAS is usually specific to deep learning models to eliminate the pitfalls of handcrafted trial-and-error design by domain experts, which doesn’t mean we have explored the entire network architecture space and achieved the best option yet. There are three main categories for searching including: Reinforcement Learning-based [16], [64], Evolutionary Algorithms-based [35], and Gradient-based [65], [66] approaches. NAS has discovered new architectures that are faster, less computationally expensive and more accurate than existing ones for many tasks. Ghiasi et al. showed NAS could discover a network architecture that surpasses the exemplar Mask R-CNN detection accuracy with less computation time [67]. Howard et al. employed NAS to search for a new MobileNetv3 architecture, which is more accurate on ImageNet classification while reducing latency by 20% compared to MobileNetV2 [59]. NAS has also discovered more accurate and lower complexity models for segmentation [69] and object tracking [70].

Hardware Acceleration

As alluded to earlier, achieving speed enhancements of neural networks require hardware-level considerations. As such, strides have been taken recently in neural network acceleration techniques to achieve improved throughput, memory overhead and energy consumption. There are two main ways of doing this, either through pure hardware optimisation or through software-hardware co-design that enables the compression techniques above to realise performance enhancement [30].

Neural networks primarily consist of two main types of layers, convolutional (Conv) and fully connected (FC) layers. While the matrix-vector multiplication (MVM) of FC layers cannot be reused, the Conv layers inherently offer an opportunity for data reusability. Modern neural network accelerators exploit this principle to optimise throughput, energy and area which are severely constrained in edge applications.

Pure hardware acceleration techniques involve the use of parallel compute units and processing-in-memory (PIM) architectures. Parallel compute units are implemented using systolic array designs that use a series of processing elements (PE) to cascade the computations across several stages. Each PE stores a local buffer of data, input activation, weights and partial sums from its multiply-accumulate (MAC) operations which are then passed to the next PE in the next cycle, thereby reducing the DRAM access time from the global buffer. PIM architectures, on the other hand, seek to accelerate neural networks by performing their computations in analog circuitry by making use of emerging non-volatile memory (eNVM) technology.

Although pure hardware acceleration provides some benefits, they quickly hit a performance wall as parallelism and data reuse opportunities are exhausted. Therefore, software-hardware co-design is now a primary focus for achieving further acceleration. The two main compression techniques that have been studied in consideration of hardware acceleration are quantisation and pruning. However, other compression methods such as Knowledge Distillation (KD) and low-rank factorisation also offer opportunities for similar enhancements through an effective hardware-software codesign framework. While model compression techniques have been discussed at an algorithmic level, below we discuss how these algorithms practically achieve speed enhancements at a hardware level.

Quantisation with fixed point representations can generally be achieved by replacing higher-precision adders and multipliers with corresponding lower-precision ones to execute the MAC operations. At the very extreme of binary or ternary quantisation, the costly MAC operations can be replaced with simpler accumulations using XNOR and pop-count logic operations [30]. However, for variable bit-width quantisation, alternative bit-serial and bit-decomposed MAC processing techniques are applied. In these approaches, MAC operations are computed using a combination of AND gates and shifted accumulations [30].

Pruning can effectively lead to weight sparsity, input sparsity and output sparsity, all of which can be exploited for hardware accelerators [30]. An early method of leveraging weight sparsity was to simply skip MAC operations with zero weights. More sophisticated techniques such as Cambricon-X include the use of indexing to access only the required activation in each PE [30]. However, indexing at a weight-level can introduce substantial overheads. Weight-vector sparsity can help overcome this issue by pruning at a channel or layer level, which helps to reduce the indexing overhead when handling pruned networks at a hardware level.

Unlike quantisation and pruning which require hardware level considerations, tensor decomposition and compactly designed models using KD or NAS can usually achieve performance enhancements directly on general-purpose processors. However, there are some edge cases where certain abnormal operations are induced as a result of tensor decomposition, which must be properly handled to avoid performance losses.

3.3 Edge Deployment Frameworks

To deploy ML using edge devices, one must prepare the model with a compatible framework that is supported by the target device. Companies like Google, Facebook, Apple, Nvidia and ARM have developed frameworks such as Tensorflow Lite, ML Kit for Firebase, PyTorch Mobile, Core ML, Tensor RT, CMSIS-NN, Embedded Learning Library (ELL) and Apache MXNet, among others, which help facilitate the deployment of ML systems to a range of different classes of edge computing hardware.

The aforementioned frameworks are typically written for specific operating systems that must be supported by the target device. Popular edge devices often run operating systems such as Linux, Android, iOS, Windows and MacOS, which provide an environment for the inference engines to perform their computations. The supported operating systems for each framework are summarised in Table 2. Some of the frameworks are also targeted for inference using bare-metal microcontrollers, which are also captured for awareness.

Table 2: Summary of framework OS requirements

Framework	Languages	Operating System					
		Linux	Android	iOS	Windows	MacOS	Bare-metal
Tensorflow Lite	C++, Java, Swift, Python	✓	✓	✓			✓
ML Kit for Firebase	C++, Java		✓	✓			
PyTorch Mobile	Python, C++, Java	✓	✓	✓			
Core ML	Swift, Python			✓			
Tensor RT	C++, Python	✓					
Jax	Python	✓			✓	✓	
Embedded Learning Library	C++, Python	✓			✓	✓	
Apache MXNet	C++, R, Python	✓			✓	✓	
CMSIS-NN	C, C++						✓
X-Cube-AI	C, C++						✓

Targeted towards edge deployment, many of these frameworks come packaged with extensive libraries of pretrained and customisable models for performing various tasks such as object recognition, landmark detection, and text recognition among others [71]. Moreover, some of these frameworks also come packaged with optimisation pipelines including quantisation, pruning, tensor fusion and kernel auto-tuning to maximise the model's performance on the target platform [10]. Two of the most popular frameworks, Tensorflow Lite and TensorRT, are described in detail below.

Tensorflow Lite

Tensorflow Lite (TF-Lite) is a lightweight ML framework developed by Google for the purpose of deploying models to mobile and edge devices as well as embedded systems. Due to its open-source licensing and cross-platform compatibility, TFLite is regarded as one of the most popular edge-ML frameworks. TF-Lite supports custom-trained models while also providing a library of pre-trained CNNs for image classification, object detection, pose estimation, text classification etc. In literature, it has been applied in image classification by Leong et al. [11] for on-orbit cloud detection and object detection by Campoverde et al. [72] for urban mobility monitoring among several other applications. The Space, High-performance and Resilient Computing (SHREC) Center at The University of Pittsburgh have been developing their own radiation-hardened SoC with a Xilinx Zynq-7020 processor, capable of running TF-Lite models for terrain classification [7].

[10] https://developer.nvidia.com/tensorrt

A core component of the TF-Lite framework includes the interpreter that is responsible for executing the TF-Lite model. To create a TF-Lite model, a pre-trained model is first passed through a model compression pipeline which performs quantisation, pruning and weight clustering to reduce the size and complexity of the model, similar to the Deep Compression framework proposed by Han et al. [73]. Following this, the inference engine applies hardware acceleration by making use of delegates to perform parallelised computations on GPUs, DSPs and Edge TPUs. A performance evaluation paper by Zhang et al. [74] demonstrated TF-Lite outperforming TensorFlow, Caffe, MXNet, PyTorch in terms of latency, memory requirement and energy consumption across a range of edge devices.

TensorRT

TensorRT is a software development kit (SDK) used for targeted acceleration onboard Nvidia GPU devices. Compared to CPU-only inference, TensorRT's high-performance deep learning inference capability provides up to a 40x boost in performance. The SDK is built on top of CUDA and provides additional device-level optimisations, in addition to the standard model compression pipelines, making it one of the most high-performing, low-latency inference engines in the market. These include kernel auto-tuning, which automatically determines the most optimal kernel configurations for each individual target device, as well as dynamic tensor memory for more efficient memory re-use capabilities. TensorRT also features tensor fusion which combines the computations of multiple layers into one to reduce the overhead of reading and writing tensor data for each layer.

Dai et al. [75] deployed robot environment sensing models onboard a Nvidia Jetson Xavier NX. The authors applied TensorRT optimisations to help achieve 2.5 times faster inference than regular deployment, enabling real-time reasoning speed. Jeong et al. [76] proposed a multi-device parallelisation for object detection using the TensorRT framework and demonstrated between 81% to 391% throughput improvement over six real-life benchmarks.

3.4 Centralised vs Decentralised vs Federated Learning

While inference at the edge has been the focus so far, it is worth noting that training at the edge can also yield several benefits. While a traditional training process involves a centralised server in the cloud that trains on data collected from its connected IoT devices, it exposes end users to the risk of data leaks. Distributed on-device learning is a paradigm whereby each IoT device is responsible for training its own model using its own collected data [77]. Finally, federated learning is another strategy whereby each device trains its own model and then shares the parameters with a central server for aggregation [77]. This enables all connected devices to benefit from the knowledge of other devices, while removing the need for data to leave the devices.

While distributed on-device learning and federated learning provide the benefit of increased data security, implementation of such training techniques can be quite a challenge due to the higher memory requirements and computationally expensive backpropagation algorithm. Nevertheless, application of federated learning for satellite swarms can enable each satellite's experiences to be shared across the network [78], which, for example, could be beneficial in learning more effective collision avoidance manoeuvres trialled by other satellites.

4. Onboard ML Platforms for Space

The ML and DL methods previously described ultimately rely on hardware platforms for execution. This section provides an overview of the computing hardware for deploying ML which have been or can potentially be used in space. It also discusses the challenges associated with doing computing in space and some of the strategies for circumventing these challenges.

4.1 Hardware-related Challenges

Some of the challenges in operating computing hardware in space include extreme radiation, extreme temperature, power and size constraints. The effects of power and size constraints limiting the computing resources and the relevant mitigation strategies have largely been discussed in Section 3. Therefore, extreme radiation and temperature are the focus of the discussion in this section.

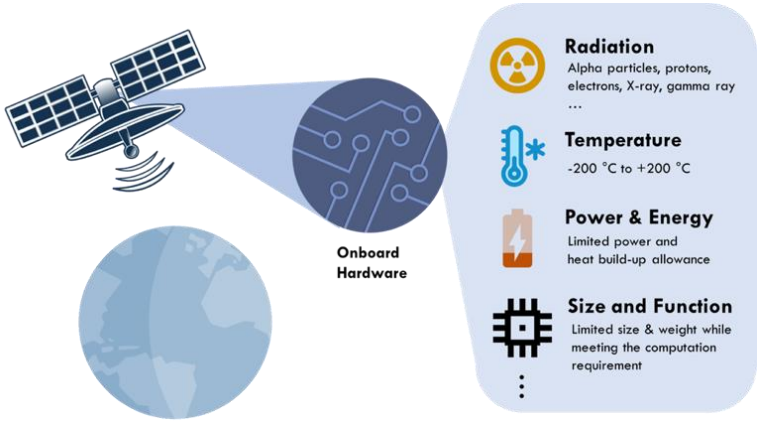

Figure 3: Challenges of operating computing hardware in a space environment are numerous. The main one is extreme radiation that interferes with computations. Temperature is another consideration that can cause electronics to melt or freeze up. Power availability is a significant challenge as satellites generate their own power using solar panels, which can be limited in capability due to the volume-constrained environment. Finally, due to the small form factor of deployable computing hardware, they tend to have lower processing power.

Extreme Radiation

Extreme ionising radiation is one of the most critical challenges to on-orbit computing platforms [79]. There are three main sources of radiation that impact electronics in space. The first source is galactic cosmic rays, which consist of electrons, photons or neutrons from outside the solar system. Secondly, radiation can originate from within the solar system, mainly from the Sun due to flares and explosions, known as high energy solar radiations. These primary and secondary particles containing electrons and ions can be captured by and trapped around a planet's magnetosphere, resulting in a belt-shaped particle cluster. This forms the third type of radiation, known as a radiation belt. There are two such belts for Earth known as the Van Allen radiation belt [80].

Without proper protection, a circuit can be influenced by any of these three forms of radiation. At the low level, the ionising radiation may result in ionisation in circuitry which can cause register states to be changed, causing what are known as bit flips or soft errors. If a particle has very high energy, the collisions can affect the arrangement of atoms in the crystal lattice of the computing chip, which will lead to permanent damages. At a higher level, these effects may be manifested as temporary malfunctions, corrupted data, permanent damage or even a complete shutdown, as radiation level increases. A device is said to be ELDRS (Enhanced Low Dose Radiation Sensitivity) free if no damages are found below a dose rate of 0.01 rad (Si)/s, which is associated with low dose rate acceptance tests.

Due to these concerns, electronics are often designed with shielding against radiation, through a series of radiation hardening techniques [81]. One way to harden the device is to manufacture on insulated materials, or to apply a physical shielding of the radiation beams [82]. This protects the device from high energy beams that can cause bit flips, effectively corrupting memory and computations, or lattice displacements causing damage to the computing chip. However, it must be noted that radiation-hardened hardware tends to be very expensive and adds significant weight to the system. Moreover, such hardware also only tends to be supported by very niche software toolchains and is therefore much more difficult to source developers for. Therefore, due to the overhead, radiation-hardened hardware is only suitable for mission critical computations, such as flight avionics and navigation systems.

On the other hand, for less mission critical applications, radiation-tolerant designs may be preferred over radiation hardened ones. One major benefit of radiation-tolerant designs is that off-the-shelf computing hardware can be used, rather than specialised space-grade hardware. Instead of applying physical protections, radiation-tolerant designs aim to account for data corruptions and correcting them at the logical level. One way is to apply parity checks to detect bit flips and correct for them [83]. Another radiation-tolerant design applied in SpaceX's Dragon spacecraft is triple modular redundancy (TMR), whereby multiple CPUs are made to perform the same computation to enable the system to perform self-checking and correcting in case of soft errors [84]. However, one downside is that these mitigation methods are only suitable for the temporary errors generated by the ionisation process.

Extreme Temperature

Temperatures below -200°C or in excess of 200°C are common occurrences in the space environment, due to the lack of atmospheric insulation that maintains the temperate conditions here on Earth. Extreme high temperatures are experienced when satellites are exposed directly to the sun's rays and extreme low temperatures are faced when they are in the shadow of the Earth. Therefore, thermal control is an essential aspect of satellite platforms for both their physical integrity and protection of their electronic equipment [11].

Typical mechanisms for thermal control include the use of physical shutters or heat pipes. However, these techniques are severely power hungry and can quickly deplete onboard power reserves [12]. Therefore, Wu et al. [85] developed a new hybrid material using silicon and vanadium dioxide that serves as a textured skin to maintain internal temperatures, similar to homeostasis in humans. Materials engineering holds the solution for effective thermal protection systems in platforms deployed to space. This is testified by NASA's Parker Solar Probe which, despite being in a 1400°C environment near the sun, maintains just above room temperature using a carbon composite shielding mechanism [13].

4.2 Hardware Platform Options

Single-board Computers

The most straightforward way to deploy an ML model is to execute it on a personal computer (PC). However, a typical desktop or laptop computer may be oversized for satellites. Single-board computers can be manufactured at a miniature size and operated at a low power. As general-purpose computers, they have operating systems that can support most mainstream deep learning libraries like TensorFlow or PyTorch. Once the model is finished training, it can be loaded on the single-board computer with minimal additional setup, which is a major benefit in using single-board computers. The disadvantage, however, mainly lies in their use of serial processing CPUs which makes it challenging for them to handle computationally intensive tasks.

[11] https://www.esa.int/Enabling_Support/Space_Engineering_Technology/Thermal_Control
[12] https://news.usc.edu/133090/why-do-usc-engineers-try-to-keep-a-satellite-warm/
[13] https://www.nasa.gov/content/goddard/parker-solar-probe-humanity-s-first-visit-to-a-star

The Raspberry Pi is a single-board computer, manufactured by the Raspberry Pi Foundation. It has an operating system, Raspbian, which is based on the Debian Linux distribution. In addition to standard computer parts, Raspberry Pi comes with a set of general-purpose input output (GPIO) pins for extensibility. In July 2019, a Raspberry Pi Zero was sent to space on the DoT-1 satellite to demonstrate the visual capturing capability [86]. This single-board computer was equipped with a set of camera modules and video of the earth was successfully captured. The computer was shielded by a metal case to avoid space radiation.

Microcontrollers

Microcontroller units (MCUs) are integrated microcomputers on a chip, where programs can be loaded and executed. User-defined code can be run without having an operating system. This adds to the efficiency and reduces the power consumption compared to a single-board computer. However, this comes with a cost that the algorithm trained for a different MCU may require rewriting using the supported language and using the correct pin references for the particular chip. This is because the development environment may not be set up on the device out of the box.

Early spacecraft were equipped with microprocessors, which are similar to microcontrollers without the integrated memory and IO units. This includes the RCA 1802 in the Galileo Jupiter Mission (1989) and the AMD 2900 for its altitude and articulation control system [87]. Modern microcontrollers such as the Arduino offer the opportunity for affordable prototyping and deployment. Arduino is an opensource microcontroller that allows users to program using embedded C++. With readily available expansion boards, known as shields, real-time data collection and processing can be setup within a very short time. Based on the Arduino, ArduSat is a nanosatellite design which is equipped with a wide range of sensors, including a camera [88]. Images of Earth taken by ArduSat were successfully transmitted back but no specific ML applications were reported with a microcontroller. However, there is no reason why a machine learning model could not be deployed from a technical point of view. Libraries for implementing machine learning and deep learning models are available, such as Perceptron [89] for Arduinos, X-Cube-AI for STM32 MCUs and CMSIS-NN for generic ARM Cortex-M processors. Although it may take some effort to convert large networks into a deployable form, microcontrollers provide a full package to deploy the ML with high robustness. As for radiation stability, it was found that after exposure to a full year's ionising radiation in medium earth orbit (MEO), no pin failure was reported in a diagnostic study. However, an error occurred when uploading code to Arduino after 186 krad of exposure [90].

Graphical Processing Units (GPU)

A graphics processing unit (GPU) is a circuit designed to manipulate graphics and images with a highly parallelised computing structure. It was originally designed for graphical processing and gaming. Since a lot of DL computations are inherently parallelisable, GPUs are more efficient compared to CPUs and have been widely adopted in training deep neural networks. In fact, growing GPU capability is the very reason for the rise of DL over the past decade. Popular options for GPUs include the Nvidia RTX series, e.g. RTX 3090, and AMD Radeon RX series, e.g. RX 6800, 6900 XT. Nvidia also provides the CUDA parallel computing framework for GPU programming, which is the interface for the DL frameworks, such as TensorFlow and PyTorch, to communicate with GPUs. In summary, GPUs are ideal devices for DL practices on the ground. However, there are still obstacles to overcome to be able to use them in space.

Typically, a GPU is a co-processor that receives tasks from a CPU and its main function is to accelerate data processing. Therefore, it is typically run alongside a CPU rather than completely on its own. Popular examples of such GPU-integrated systems on a chip (SoC) include the Jetson Nano, TX1, TX2 and Xavier platforms [14]. However, GPU-only systems are far more favourable for space applications. Therefore, a special type of GPU has been developed called general purpose GPUs (GPGPUs), which can be run independently in specialised embedded systems. The Intel Xeon Phi is a GPGPU with certain

[14] https://www.nvidia.com/en-au/autonomous-machines/jetson-store/

versions able to run as a primary processor in a Linux-based operating system [91]. It stemmed from the Larrabee project and was mainly designed for high performance computing and ML use cases.

Despite there being no existing reports on employing GPUs alone in space, research has been widely carried out to test their feasibility. Government and national agencies play an important role in setting benchmarks and performing tests. Based on a NASA technical report, GPUs and GPGPUs are tested for a range of ML-centric tasks including scientific sensing, object tracking, obstacle identification, neural network convergence, image processing and data compression [92]. Controlled radiation experiments were conducted using devices such as Nvidia TX1 SoC, GTX 1050, Intel Skylake and AMD RX460. Measurements were reported on key radiation characteristics such as the cross-section and flux of the proton particles. Similarly, academic literature has been actively searching for possible GPU-based solutions for space missions. Adams et al. [93] developed a high-performance on-orbit computing system by integrating a traditional flight computer with an existing GPU, the Nvidia Tegra X2/X2i.

Application Specific Integrated Circuits (ASIC)

Application specific integrated circuits (ASICs) are integrated circuits designed for a specific application. Modern ASICs, that typically include a microprocessor, memory and other blocks, are often referred as a system-on-a-chip (SoC). Different from computers or microcontrollers, ASICs are designed for specific tasks. Radiation is once again the main concern for ASICs being used in space as they can deliver permanent circuit damage. This can result in mission failure as it is not an option to replace or fix an onboard computing chip in space. As a result, a major effort towards space-grade devices lies in the design of radiation-hardened ASICs. In a proposal by NASA, the tolerance of radiation-hardened ASICs should be at least 1 Mrad total ionising dose, with a latch up immune to a linear energy transfer of at least 80 dE/dx [94]. Although such ASICs are currently mainly provided by commercial vendors, with the growing chip design capability of SpaceX, in-house production may not be too distant in the future.

ASICs come in various forms depending on the task to be performed. Two commonly used ASICs for edge ML are the vision processing unit (VPU) and the tensor processing unit (TPU). VPUs are microprocessors that accelerate computer vision related tasks, primarily involving convolutional layers. Last year's PhiSat-1 (Φ-Sat-1) launch with onboard ML featured Intel's Movidius Myria 2 VPU. The Myriad 2 VPU is a high-performance unit with specific optimisations for low power consumption. With the descendant Myriad X, both chips come in the form of USB sticks, e.g., Intel Neural Compute Stick 2, and are widely used in drones [95]. Unlike VPUs, there is no current reported launch of TPUs in space. TPUs are devices developed by Google specifically optimised for neural networks, which are highly suited to deploy deep learning models. Despite a lack of launched cases, several studies were reported on the design of TPUs for satellites. Goodwill et al. investigated the design and capabilities of a CubeSat-sized Edge TPU-based co-processor card, known as the SpaceCube low-power edge AI resilient node [96]. This design conforms to CubeSat specifications for integration into next-generation SmallSat and CubeSat systems, both of which are required by NASA on miniaturised satellites [97].

Field Programmable Gate Arrays

A field-programmable gate array (FPGA) is an integrated circuit that can be programmed for different tasks with re-configurable gate-level control. It differs from ASICs in that ASICs perform a fixed task throughout the life cycle, while FPGA can be reconfigured. In general, ASICs are more suited to achieve high performance for a specific unchanged task, while FPGAs are used for more flexible missions for their re-programmability. With an increased satellite lifespan, re-programmability in flight will gradually become a requirement. This leads to the increasing use of high-performance FPGAs in space in place of ASICs. For example, in a European Space Agency funded project CloudScout, Rapuano et. al. used a FPGA to accelerate a CNN model which led to a reduced inference time, and a higher possibility of customisation, but at the cost of greater power consumption and a longer Time to Market [98].

The challenge of operating FPGAs in space also surrounds radiation. The FPGAs suitable for space have been previously based on static random-access memory (SRAM), which is the component used

to switch the transistor state. While these FPGAs provide flexibility due to the re-programmability, any radiation upsets in the SRAM can cause the underlying configuration to be modified and lead to a system malfunction. As a result, FPGAs should be radiation hardened to suit space missions. Measures can be taken both at the hardware level, e.g., shielding and redundancy computing, or the software level, e.g., rollback recovery and error detection, to mitigate the radiation effects. Vendors like Xilinx improve the radiation tolerance with methods such as imbalanced latches for configuration circuitry. Xilinx's Virtex II QPro and Virtex II Pro are popular choices for space applications. Other radiation-hardened FPGA products include the RTG4 from Microchip. Another approach to address radiation instability is a flash-based FPGA, which uses non-volatile memory cells. Speers et al. [99] compared a flash-based solution with a SRAM-based FPGA, which proved the former to be superior for single-event gate rupture and for mitigating functional upsets.

4.3 Software-Hardware Integration

Hardware Optimisations for ML Models

Extensive work has been performed to tailor the hardware for running ML algorithms [91], [100]. Speed is one of the most important aspects to optimise which pushes the boundary of hardware acceleration. In recent years, FPGAs have shown a great potential for algorithm acceleration [101]. A strong emphasis is made on CNNs as they are the most widely used DL model for image related work. In a typical FPGA-accelerated system, an FPGA works with a host CPU. Each has its own memory while the memory can be shared. The FPGA can work simultaneously with the CPU on different parts of the shared memory. To further improve the training efficiency, binary neural networks (BNNs) are often used [102]. In a neural network like a CNN, there are usually millions of parameters stored as floating-point numbers. A BNN is a remake of a conventional neural network so that each parameter is binarised, without significant changes to the overall performance. A BNN reduces the memory and storage from 32 bit to 1 bit per parameter. In addition, the binarisation of a model enables the neural network to be expressed in the form of additions or subtractions, which are much less expensive operations compared with multiplication. Optimised FPGAs can reach a comparable performance level to GPUs. Examples of such products include the BittWare Stratix 10 based on the Intel FPGA OpenCL compiler [103]. Besides speed, optimisations are reported on the energy efficiency and power consumption. A number of excellent reviews are available on this topic [56], [104], [105].

ML Performance on Hardware Platforms

To put the effects of all the edge processing techniques discussed into context, it is worth presenting a results table of the different algorithms across a variety of different devices. This is shown in Table 3, outlining the latency of image classification and object detection tasks using the ResNet and SSD models respectively using various types of platforms and accelerators.

The results clearly demonstrate the advantage of using accelerators for ML workloads compared to CPU-only setups. In fact, greater than 30x speedup over CPU-only systems can be achieved even when using a relatively low-power GPU, such as the Jetson Xavier NX. However, the use a GPU alone is not enough to achieve the speedups demonstrated here. It is worth noting that an attributable factor to this significant speedup is the use of the TensorRT inference engine which is highly specialised for NVIDIA hardware.

Table 3: Performance Summary [15]

System	Processor	Accelerator	Software	Latency (ms)	
				Classification (ImageNet, ResNet)	Object Detection (COCO, SSD-small)
Raspberry Pi 4	Arm Contex-A72 MP4	-	TFLite v2.5.0 (ruy)	513.37	-
Amazon EC2	Neoverse-N1	-	TVM v0.8-dev	113.35	-
NVIDIA Jetson AGX Xavier	NVIDIA Carmel	-	ArmNN v21.05 (Neon)	72.08	-
NVIDIA Jetson Xavier NX	NVIDIA Carmel	NVIDIA Xavier NX	Jetpack 4.6, TensorRT 8.0., CUDA 10.2	2.44	1.36
Lenovo ThingSystem SE350 server	Intel® Xeon® D-2123IT CPU @ 2.20GHz	NVIDIA T4	TensorRT 8.0.2, CUDA 11.3	0.82	0.47
NVIDIA DGX A100	AMD EPYC 7742	NVIDIA A100-SXM-80GB	TensorRT 8.0.1, CUDA 11.3	0.66	0.44
Dell EMC PowerEdge XE2420	Intel® Xeon® Gold 6252N CPU @ 2.30 GHz	NVIDIA A10	TensorRT 8.0.2, CUDA 11.3	0.46	-

4.4 Tools, Frameworks and Platforms

Hardware Related Tools

Automation tools are available for hardware. For example, automation tools can generate the design in hardware design language (HDL) for FPGA based on the network structure [106]. The intention is to ensure the hardware can always achieve optimal performance on the target platform. This approach suits when the network structure is mostly static without frequent updates [104].

Software Frameworks

Apart from the general frameworks mentioned earlier, a large selection of frameworks are available for DL regarding hardware integration. Most frameworks are built based on C / C++ based frameworks such as the Caffe library. For example, DNNWeaver is a framework built on Caffe which takes a pair of DL and FPGA configuration files and generates synthesisable deigns for a variety of FPGA platforms [107]. Caffeinated FPGAs is an extension of Caffe on Xilinx SDAccel for CNN to be run on CPU-FPGA systems [108]. Other similar frameworks such as AI2GO, DeepThings, DeepCham, DeepX and EdgeML provide

[15] https://mlcommons.org/en/inference-edge-11/

a rich set of tools for developers [4]. Among them, some provide support for BNN. FINN is a framework to build BNN inference accelerators on FPGAs. Nvidia has released several tools on GPU. In addition to TensorRT, RAPIDS is another Nvidia GPU-accelerator for model training with a speed up factor of 15.6 [109]. Intel released the Intel Math Kernel Library for DNN, which is a library to implement CNNs with C/C++ on Intel CPU and GPUs [110]. Intel has also distributed its Caffe framework known as the IntelCaffe for performance improvement on Intel chipsets [111].

Platforms

As space ML is still in its infancy, the reports on platforms are quite limited. An Australian company named Aurora is developing a distributed FlatSat to facilitate pre-flight qualification of small satellites [112]. A conventional FlatSat is a set of electronics modules connected by wires to simulate actions in space before the construction of the physical parts. A distributed FlatSat extends this idea by incorporating remote modules, which offers a good playground to accommodate edge computing and federated learning.

5. Challenges, Opportunities and Trends

5.1 Challenges

Environmental Constraints

The unique environment in space imposes a number of constraints on the hardware. Due to the limited size of satellite platforms, the overall configuration of the hardware is also limited. This includes the memory space, the processing capability and the power consumption. Some radiation-tolerant designs, such as the redundant computing, add weight and volume to the system. Other factors such as hardware acceleration also influences the physical size of the device. These should be taken into careful consideration at an architectural level to balance the size and performance. In addition, the harsh environment in space set other criteria on the hardware. Apart from the high radiation described previously, the extreme temperature requires the hardware to fully operate across a wide temperature range from –200°C to +200°C. Other factors include vibrations from pyrotechnic shocks, the release of gases, known as outgassing, high levels of static discharge and being in a vacuum. However, testing always plays an important role at the end of prototyping to assess the hardware robustness. A common test to perform is the low dose rate acceptance assessment based on the ELDRS standards [113].

Data Security

Data security should be taken into account during data transmission. This can be sending the data back to the ground in the traditional way or by sharing model parameters via edge computing. Fortunately, there are various layers to control access and mitigate potential data breaches. First, essential privacy information can be de-identified from the source. This pre-processing step can be performed by the onboard algorithm after data acquisition. Secondly, encryption can be applied to avoid data being intercepted during transmission. Encryption methods based on key authentications can be utilised. In a typical asymmetric key encryption procedure, the data is encrypted by a public key. The encrypted information is transmitted rather than the original data. The encrypted data can only be decrypted using a corresponding private key, which is generated with the public key and held by the receivers [114]. As most of the space ML is surrounding image data, it is of particular interest to be aware of the encryption techniques on images [115]. This approach is highly suitable for data sensitive and commercial applications. As another security layer, a credit system can be established for the data consumers. Users with a bad credit record should be excluded to minimise the likelihood of data misuse.

Legal Considerations

ML in space also brings thoughts regarding legal considerations. As ML is a fast-developing area, constant adjustments to the legislation can be difficult. Hence very little work has been done in this regard, despite space law being a well-established discipline [116]. In March 2012, the RoboLaw project was launched to focus on the issues brought by the legal status of the ML ecosystem such as robotics

and computer-brain interfaces [117]. The awareness of privacy and potential influence of ML increased since 2018, when ML started to demonstrate a close-to-human performance in many areas. Legislation and declarations were made on a regional basis, including California, Toronto and Montreal [118]. It can be foreseen that with further development of space ML, more concerns will be raised with legislation refinements. For example, privacy might arise at a regional level. A city might want certain areas to be de-identified from a satellite image, just like a person might want the face de-identified from a Google car. This can be an act to protect sensitive information such as military and commercial secrets. It will be a coadoption of multiple factors, including space ML, legislation, regions and individuals. We are currently witnessing this trend at a domestic level but international consensus could be some time away.

5.2 Opportunities and Trends

Regarding the hardware, the physical size of the circuitry is generally getting smaller. This is evident not only for space related circuits, but also for broader scenarios such as edge computing and IoT. A reduced size brings advantages including improved mobility and lower power consumption. Secondly, the integration between hardware and software has drawn growing attention, which is particularly common in the development of FPGAs and GPUs. This has resulted in an enrichment of tool sets and design patterns such as software-aware hardware designs or hardware-aware NAS [101]. Hardware acceleration can be better achieved with an improved software-hardware interface, which makes up for the computing power of small-size circuits. Thirdly, radiation hardening is the main topic for most hardware for space applications. Current techniques such as physical shielding and soft error correction are very mature and well tested. With that as a general framework, future work can be developing techniques for specific parts of the circuit and providing further enhancements to the more vulnerable parts, such as the SRAM on FPGAs. Fourthly, with rapidly improving camera technologies and increasing ground sample distance (GSD) of satellite imagery, a wide range of new space applications will become feasible, which will require on-orbit ML solutions to operate. Finally, with the rise of quantum computing, it is already seeing itself deployed for space-based quantum communications [16]. If the use of quantum computing in space takes over, it will usher in a new era of quantum space ML, enabling significantly faster processing than classical computers.

6. Conclusion

Machine learning has the potential to play a key role in the rapidly growing space industry. Applications ranging from earth observation, communications, navigation and fault detection, isolation and recovery (FDIR) involve huge amounts of data that need to be processed in real time by a computer. This report presented why it is imperative to process collected data using available flight computers, rather than downlinking data to ground stations for processing. Fortunately, the recent miniaturisation of edge devices has made it feasible to deploy ML-capable hardware to space onboard various classes of satellites, including CubeSats.

A comprehensive study into the available edge platforms for space applications revealed a range of devices including SoCs, FPGAs and ASICs. However, the choice of hardware is largely determined by the computational demand of the algorithm, which in turn is dictated by the application. Therefore, the report touches on important applications that necessarily require real-time processing, energy usage and improved autonomy. A deep dive into the algorithms revealed several efficiency-focused ML models for general edge computing applications, which can potentially be adopted for space applications. Despite the presence of some efficient models, our performance comparison showed that their performance generally tends to be lower than their larger counterparts. This led to a study of model compression and hardware acceleration techniques that target efficient processing with maximum

[16] https://www.scientificamerican.com/article/china-reaches-new-milestonein-space-based-quantum-communications/

retention of performance. With extensive framework support, these techniques show promise for deploying ML models to space using low resource computing platforms.

However, aside from being compute and power constrained, one of the main challenges in deploying ML to space is the harsh space environment. For example, extreme radiation can cause bit flips, effectively corrupting computations. Radiation hardening is the technique that is used to shield critical electronics from radiation. However, this process can add significant weight and cost to the mission and therefore is only applied for mission critical applications. An alternate technique called redundant computing is often used to perform self-checking to correct computations in the event of a bit flip.

In conclusion, it can be seen that the technology, hardware and community are ready to apply ML to deliver value to various space applications. Rapidly improving camera technologies and the rise of quantum computing are expected to usher in a new era of on-orbit ML computing. However, various environmental constraints, data security issues and legal considerations have insofar prevented a widescale adoption of ML for space applications. Therefore, once these concerns are alleviated, perhaps through the use of on-orbit computing, it is hoped that space ML will be positively leveraged by businesses to benefit society.

7. References

[1] T. Hoeser and C. Kuenzer, “Object Detection and Image Segmentation with Deep Learning on Earth Observation Data: A Review-Part I: Evolution and Recent Trends,” *Remote Sens.*, vol. 12, no. 10, 2020.

[2] F. Fourati and M.-S. Alouini, “Artificial Intelligence for Satellite Communication: A Review,” *ArXiv*, vol. abs/2101.10899, 2021.

[3] V. Kothari, E. Liberis, and N. D. Lane, “The Final Frontier: Deep Learning in Space.” 2020.

[4] M. Lofqvist and J. Cano, “Accelerating Deep Learning Applications in Space.” 2020.

[5] X. Feng, Y. Jiang, X. Yang, M. Du, and X. Li, “Computer vision algorithms and hardware implementations: A survey,” *Integration*, vol. 69, pp. 309–320, 2019.

[6] M. G. S. Murshed, C. Murphy, D. Hou, N. Khan, G. Ananthanarayanan, and F. Hussain, “Machine Learning at the Network Edge: A Survey,” *ACM Comput. Surv.*, vol. 54, no. 8, pp. 1–37, Nov. 2022, doi: 10.1145/3469029.

[7] J. Manning *et al.*, “Machine-learning space applications on smallsat platforms with tensorflow,” 2018.

[8] B. Denby and B. Lucia, “Orbital Edge Computing: Machine Inference in Space,” *IEEE Comput. Archit. Lett.*, vol. 18, no. 1, pp. 59–62, Jan. 2019, doi: 10.1109/LCA.2019.2907539.

[9] A. Guiotto, A. Martelli, and C. Paccagnini, “SMART-FDIR: Use of Artificial Intelligence in the implementation of a Satellite FDIR,” *Eur. Space Agency Spec. Publ. ESA SP*, vol. 532, p. 71, Jan. 2003.

[10] A. S. Kucik and G. Meoni, “Investigating Spiking Neural Networks for Energy-Efficient On-Board AI Applications. A Case Study in Land Cover and Land Use Classification,” in *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, 2021, pp. 2020–2030. doi: 10.1109/CVPRW53098.2021.00230.

[11] T. I. Leong, Y. M. O. Abbas, M. A. C. Purio, and H. A. Elmegharbel, “Image Classification Unit: A U-Net Convolutional Neural Network for On-Orbit Cloud Detection Aboard CubeSats,” in *2021 IEEE International Geoscience and Remote Sensing Symposium IGARSS*, 2021, pp. 2807–2810. doi: 10.1109/IGARSS47720.2021.9553156.

[12] K. Simonyan and A. Zisserman, “Very deep convolutional networks for large-scale image recognition,” San Diego, CA, United states, 2015.

[13] C. Szegedy, S. Ioffe, V. Vanhoucke, and A. A. Alemi, “Inception-v4, inception-ResNet and the impact of residual connections on learning,” in *31st AAAI Conference on Artificial Intelligence, AAAI 2017*, San Francisco, CA, United states, 2017, pp. 4278–4284.

[14] K. He, X. Zhang, S. Ren, and J. Sun, “Deep Residual Learning for Image Recognition,” in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 770–778. doi: 10.1109/CVPR.2016.90.

[15] M. Tan and Q. V. Le, “EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks,” 2019.

[16] J. D. Co-Reyes *et al.*, “Evolving Reinforcement Learning Algorithms,” 2021. [Online]. Available: https://openreview.net/forum?id=0XXpJ4OtjW

[17] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, “ImageNet: A Large-Scale Hierarchical Image Database,” 2009.

[18] X. Zhu and A. B. Goldberg, “Introduction to semi-supervised learning,” *Synth. Lect. Artif. Intell. Mach. Learn.*, vol. 3, no. 1, pp. 1–130, 2009.

[19] J. Gou, B. Yu, S. J. Maybank, and D. Tao, “Knowledge Distillation: A Survey,” *Int. J. Comput. Vis.*, vol. 129, no. 6, pp. 1789–1819, 2021.

[20] F. Zhuang *et al.*, “A Comprehensive Survey on Transfer Learning,” *Proc. IEEE*, vol. 109, no. 1, pp. 43–76, 2021.

[21] M. Wang and W. Deng, “Deep visual domain adaptation: a survey,” *Neurocomputing*, vol. 312, pp. 135–53, Oct. 2018.

[22] G.-S. Xia *et al.*, “DOTA: A Large-scale Dataset for Object Detection in Aerial Images,” *ArXiv171110398 Cs*, May 2019, Accessed: Nov. 29, 2021. [Online]. Available: http://arxiv.org/abs/1711.10398

[23] D. Lam *et al.*, “xView: Objects in Context in Overhead Imagery,” *ArXiv180207856 Cs*, Feb. 2018, Accessed: Nov. 29, 2021. [Online]. Available: http://arxiv.org/abs/1802.07856

[24] A. Van Etten, D. Lindenbaum, and T. M. Bacastow, “SpaceNet: A Remote Sensing Dataset and Challenge Series,” *ArXiv180701232 Cs*, Jul. 2019, Accessed: Nov. 29, 2021. [Online]. Available: http://arxiv.org/abs/1807.01232

[25] S. Ren, K. He, R. Girshick, and J. Sun, “Faster R-CNN: Towards real-time object detection with region proposal networks,” in *NeuRIPS*, 2015, pp. 91–99.

[26] Z. Cai and N. Vasconcelos, “Cascade R-CNN: Delving Into High Quality Object Detection,” in *CVPR*, 2018, pp. 6154–6162. doi: 10.1109/CVPR.2018.00644.

[27] A. Bochkovskiy, C.-Y. Wang, and H.-Y. M. Liao, “YOLOv4: Optimal Speed and Accuracy of Object Detection.” 2020.

[28] C.-Y. Wang, A. Bochkovskiy, and H.-Y. M. Liao, “Scaled-YOLOv4: Scaling Cross Stage Partial Network,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun. 2021, pp. 13029–13038.

[29] W. Liu *et al.*, “SSD: Single Shot MultiBox Detector,” in *Computer Vision – ECCV 2016*, vol. 9905, B. Leibe, J. Matas, N. Sebe, and M. Welling, Eds. Cham: Springer International Publishing, 2016, pp. 21–37. doi: 10.1007/978-3-319-46448-0_2.

[30] L. Deng, G. Li, S. Han, L. Shi, and Y. Xie, “Model Compression and Hardware Acceleration for Neural Networks: A Comprehensive Survey,” *Proc. IEEE*, vol. 108, no. 4, pp. 485–532, 2020, doi: 10.1109/JPROC.2020.2976475.

[31] T. Liang, J. Glossner, L. Wang, S. Shi, and X. Zhang, “Pruning and quantization for deep neural network acceleration: A survey,” *Neurocomputing*, vol. 461, pp. 370–403, 2021.

[32] S. Swaminathan, D. Garg, R. Kannan, and F. Andres, “Sparse low rank factorization for deep neural network compression,” *Neurocomputing*, vol. 398, pp. 185–196, 2020.

[33] G. Hinton, O. Vinyals, and J. Dean, “Distilling the Knowledge in a Neural Network.” 2015.

[34] X. He, K. Zhao, and X. Chu, “AutoML: A Survey of the State-of-the-Art,” *Knowl.-Based Syst.*, vol. 212, p. 106622, 2021.

[35] E. Real, A. Aggarwal, Y. Huang, and Q. V. Le, “Regularized evolution for image classifier architecture search,” in *AAAI Conference on Artificial Intelligence (AAAI)*, Honolulu, HI, United states, 2019, pp. 4780–4789.

[36] A. G. Howard *et al.*, “MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications [arXiv],” *arXiv*, p. 9 pp.-, Apr. 2017.

[37] X. Zhang, X. Zhou, M. Lin, and J. Sun, “ShuffleNet: An Extremely Efficient Convolutional Neural Network for Mobile Devices,” in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, Salt Lake City, UT, United states, 2018, pp. 6848–6856.

[38] R. J. Wang, X. Li, and C. X. Ling, “Pelee: A Real-Time Object Detection System on Mobile Devices,” in *Advances in Neural Information Processing Systems 31*, S. Bengio, H. Wallach, H.

Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, Eds. Curran Associates, Inc., 2018, pp. 1963–1972.

[39] J. Long, E. Shelhamer, and T. Darrell, “Fully convolutional networks for semantic segmentation,” Boston, MA, United states, 2015, vol. 07-12-June-2015, pp. 431–440.

[40] O. Ronneberger, P. Fischer, and T. Brox, “U-Net: Convolutional Networks for Biomedical Image Segmentation,” in *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015*, Cham, 2015, pp. 234–241.

[41] S. Zheng *et al.*, “Conditional random fields as recurrent neural networks,” in *2015 IEEE International Conference on Computer Vision (ICCV). Proceedings*, Los Alamitos, CA, USA, 2015, pp. 1529–37.

[42] L.-C. Chen, G. Papandreou, F. Schroff, and H. Adam, “Rethinking Atrous Convolution for Semantic Image Segmentation,” *CoRR*, vol. abs/1706.05587, 2017, [Online]. Available: http://arxiv.org/abs/1706.05587

[43] A. Kendall, V. Badrinarayanan, and R. Cipolla, “Bayesian segnet: Model uncertainty in deep convolutional encoder-decoder architectures for scene understanding,” in *British Machine Vision Conference 2017, BMVC 2017*, London, United kingdom, 2017, p. Amazon; et al.; Facebook Oculus; Microsoft Research; SCAPE; Snap-.

[44] W. Liu and F. Su, “Unsupervised Adversarial Domain Adaptation Network for Semantic Segmentation,” *IEEE Geosci. Remote Sens. Lett.*, vol. 17, no. 11, pp. 1978–1982, 2020, doi: 10.1109/LGRS.2019.2956490.

[45] Z. Lu, Z. Fu, T. Xiang, P. Han, L. Wang, and X. Gao, “Learning from Weak and Noisy Labels for Semantic Segmentation,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 39, no. 3, pp. 486–500, 2017, doi: 10.1109/TPAMI.2016.2552172.

[46] A. Price and K. Yoshida, “A Monocular Pose Estimation Case Study: The Hayabusa2 Minerva-II2 Deployment,” in *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, 2021, pp. 1992–2001. doi: 10.1109/CVPRW53098.2021.00227.

[47] B. Chen, J. Cao, A. Parra, and T.-J. Chin, “Satellite Pose Estimation with Deep Landmark Regression and Nonlinear Pose Refinement,” 2019.

[48] A. Garcia *et al.*, “LSPnet: A 2D Localization-oriented Spacecraft Pose Estimation Neural Network,” in *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, 2021, pp. 2048–2056. doi: 10.1109/CVPRW53098.2021.00233.

[49] S. Yang, Z. Quan, M. Nie, and W. Yang, “TransPose: Keypoint Localization via Transformer,” 2021.

[50] B. Artacho and A. E. Savakis, “OmniPose: A Multi-Scale Framework for Multi-Person Pose Estimation,” *CoRR*, vol. abs/2103.10180, 2021, [Online]. Available: https://arxiv.org/abs/2103.10180

[51] Z. Fan, Y. Zhu, Y. He, Q. Sun, H. Liu, and J. He, “Deep Learning on Monocular Object Pose Detection and Tracking: A Comprehensive Overview,” *CoRR*, vol. abs/2105.14291, 2021, [Online]. Available: https://arxiv.org/abs/2105.14291

[52] M. Kisantal, S. Sharma, T. H. Park, D. Izzo, M. Märtens, and S. D’Amico, “Satellite Pose Estimation Challenge: Dataset, Competition Design, and Results,” *IEEE Trans. Aerosp. Electron. Syst.*, vol. 56, no. 5, pp. 4083–4098, 2020, doi: 10.1109/TAES.2020.2989063.

[53] T. T. Pham *et al.*, “Airborne Object Detection Using Hyperspectral Imaging: Deep Learning Review,” in *Computational Science and Its Applications – ICCSA 2019*, Cham, 2019, pp. 306–321.

[54] M. J. Khan, H. S. Khan, A. Yousaf, K. Khurshid, and A. Abbas, “Modern Trends in Hyperspectral Image Analysis: A Review,” *IEEE Access*, vol. 6, pp. 14118–14129, 2018, doi: 10.1109/ACCESS.2018.2812999.

[55] G. Menghani, “Efficient Deep Learning: A Survey on Making Deep Learning Models Smaller, Faster, and Better,” *CoRR*, vol. abs/2106.08962, 2021, [Online]. Available: https://arxiv.org/abs/2106.08962

[56] J. Cheng, P. Wang, G. Li, Q. Hu, and H. Lu, “Recent Advances in Efficient Computation of Deep Convolutional Neural Networks,” *ArXiv180200939 Cs*, Feb. 2018, Accessed: Nov. 15, 2021. [Online]. Available: http://arxiv.org/abs/1802.00939

[57] T. Hoefler, D. Alistarh, T. Ben-Nun, N. Dryden, and A. Peste, “Sparsity in Deep Learning: Pruning and growth for efficient inference and training in neural networks,” 2021.

[58] N. Wang, J. Choi, D. Brand, C.-Y. Chen, and K. Gopalakrishnan, “Training Deep Neural Networks with 8-Bit Floating Point Numbers,” in *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, Red Hook, NY, USA, 2018, pp. 7686–7695.

[59] J. Yang *et al.*, “Quantization Networks,” Jun. 2019.

[60] Y. Choi, M. El-Khamy, and J. Lee, “Towards the Limit of Network Quantization.” 2017.

[61] G. Nguyen *et al.*, “Machine Learning and Deep Learning Frameworks and Libraries for Large-Scale Data Mining: A Survey,” *Artif Intell Rev*, vol. 52, no. 1, pp. 77–124, Jun. 2019.

[62] H. Yang *et al.*, “Learning Low-rank Deep Neural Networks via Singular Vector Orthogonality Regularization and Singular Value Sparsification,” in *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW). Proceedings*, Piscataway, NJ, USA, 2020, pp. 2899–908.

[63] T. Elsken, J. H. Metzen, F. Hutter, and others, “Neural architecture search: A survey.,” *J Mach Learn Res*, vol. 20, no. 55, pp. 1–21, 2019.

[64] B. Zoph and Q. V. Le, “Neural architecture search with reinforcement learning,” Toulon, France, 2017.

[65] H. Liu, K. Simonyan, and Y. Yang, “DARTS: Differentiable architecture search,” New Orleans, LA, United states, 2019.

[66] K. Nguyen, C. Fookes, and S. Sridharan, “Constrained Design of Deep Iris Networks,” *IEEE Trans. Image Process.*, vol. 29, pp. 7166–7175, 2020, doi: 10.1109/TIP.2020.2999211.

[67] G. Ghiasi, T.-Y. Lin, and Q. V. Le, “Nas-fpn: Learning scalable feature pyramid architecture for object detection,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 7036–7045.

[68] A. Howard *et al.*, “Searching for MobileNetV3,” 2019.

[69] C. Liu *et al.*, “Auto-DeepLab: Hierarchical Neural Architecture Search for Semantic Image Segmentation,” 2019.

[70] B. Yan, H. Peng, K. Wu, D. Wang, J. Fu, and H. Lu, “LightTrack: Finding Lightweight Neural Networks for Object Tracking via One-Shot Architecture Search,” 2021.

[71] Dennis, Don Kurian and Gopinath, Sridhar and Gupta, Chirag and Kumar, Ashish and Kusupati, Aditya and Patil, Shishir G and Simhadri, Harsha Vardhan, *EdgeML: Machine Learning for resource-constrained edge devices*. [Online]. Available: https://github.com/Microsoft/EdgeML

[72] A. Campoverde and G. Barros, “Detection and Classification of Urban Actors Through TensorFlow with an Android Device,” in *Information and Communication Technologies of Ecuador (TIC.EC)*, 2020.

[73] S. Han, H. Mao, and W. J. Dally, “Deep Compression: Compressing Deep Neural Networks with Pruning, Trained Quantization and Huffman Coding.” 2016.

[74] X. Zhang, Y. Wang, and W. Shi, “pCAMP: Performance Comparison of Machine Learning Packages on the Edges,” Boston, MA, Jul. 2018. [Online]. Available: https://www.usenix.org/conference/hotedge18/presentation/zhang

[75] B. Dai *et al.*, “Field Robot Environment Sensing Technology Based on TensorRT,” in *Intelligent Robotics and Applications*, 2021.

[76] E. Jeong, J. Kim, S. Tan, J. Lee, and S. Ha, “Deep Learning Inference Parallelization on Heterogeneous Processors with TensorRT,” *IEEE Embed. Syst. Lett.*, pp. 1–1, 2021, doi: 10.1109/LES.2021.3087707.

[77] S. Abdulrahman, H. Tout, H. Ould-Slimane, A. Mourad, C. Talhi, and M. Guizani, “A Survey on Federated Learning: The Journey From Centralized to Distributed On-Site Learning and Beyond,” *IEEE Internet Things J.*, vol. PP, Oct. 2020, doi: 10.1109/JIOT.2020.3030072.

[78] J. Senthilnath, S. N. Omkar, V. Mani, R. Prasad, R. Rajendra, and P. B. Shreyas, “Multi-sensor satellite remote sensing images for flood assessment using swarm intelligence,” 2015.

[79] E. G. Stassinopoulos and J. P. Raymond, “The space radiation environment for electronics,” *Proc. IEEE*, vol. 76, no. 11, pp. 1423–1442, 1988, doi: 10.1109/5.90113.

[80] Y. Chen, G. D. Reeves, and R. H. Friedel, “The energization of relativistic electrons in the outer Van Allen radiation belt,” *Nat. Phys.*, vol. 3, no. 9, pp. 614–617, 2007.

[81] W. Dawes Jr, “Radiation Effects Hardening Techniques,” Sandia National Labs., 1985.

[82] E. W. Taylor, “Organics, polymers and nanotechnology for radiation hardening and shielding applications,” in *Nanophotonics and Macrophotonics for Space Environments*, 2007, vol. 6713, p. 671307.

[83] C.-L. Chen and M. Hsiao, “Error-correcting codes for semiconductor memory applications: A state-of-the-art review,” *IBM J. Res. Dev.*, vol. 28, no. 2, pp. 124–134, 1984.

[84] T. Arifeen *et al.*, “Approximate Triple Modular Redundancy: A Survey,” *IEEE Access*, vol. 8, 2020, doi: 10.1109/ACCESS.2020.3012673.

[85] S.-H. Wu *et al.*, “Thermal homeostasis using microstructured phase-change materials,” *Optica*, vol. 4, no. 11, p. 1390, Nov. 2017, doi: 10.1364/OPTICA.4.001390.

[86] “Raspberry Pi in space! - Technology OrgTechnology Org.” Sep. 2019. Accessed: Oct. 13, 2021. [Online]. Available: https://www.technology.org/2019/09/11/raspberry-pi-in-space/

[87] J. Eickhoff, “Historic Introduction to Onboard Computers,” in *Onboard Computers, Onboard Software and Satellite Operations: An Introduction*, J. Eickhoff, Ed. Berlin, Heidelberg: Springer, 2012, pp. 21–49. doi: 10.1007/978-3-642-25170-2_3.

[88] “ArduSat,” *Wikipedia*. Jun. 2021. Accessed: Oct. 13, 2021. [Online]. Available: https://en.wikipedia.org/w/index.php?title=ArduSat&oldid=1031092454

[89] “Arduino Perceptron Library for Arduino.” Arduino Libraries, Jul. 2021. Accessed: Nov. 15, 2021. [Online]. Available: https://github.com/arduino-libraries/Arduino_Perceptron

[90] W. Olsen, “Microcontroller Survivability in Space Conditions,” p. 11.

[91] M. A. Talib, S. Majzoub, Q. Nasir, and D. Jamal, “A systematic literature review on hardware implementation of artificial intelligence algorithms,” *J. Supercomput.*, pp. 1–42, 2020.

[92] E. J. Wyrwas, “Graphics Processor Units (GPUs).” Jun. 2017. Accessed: Nov. 15, 2021. [Online]. Available: https://ntrs.nasa.gov/citations/20170006038

[93] C. Adams, A. Spain, J. Parker, M. Hevert, J. Roach, and D. Cotten, “Towards an Integrated GPU Accelerated SoC as a Flight Computer for Small Satellites,” in *2019 IEEE Aerospace Conference*, Mar. 2019, pp. 1–7. doi: 10.1109/AERO.2019.8741765.

[94] S. Buchner, “Radiation Hardness Assurance (RHA) for Space Systems,” p. 61.

[95] “Deep Neural Networks: A Comparison on Different Computing Platforms \textbar IEEE Conference Publication \textbar IEEE Xplore.” Accessed: Nov. 15, 2021. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/8575778

[96] J. Goodwill *et al.*, “NASA SpaceCube Edge TPU SmallSat Card for Autonomous Operations and Onboard Science-Data Analysis,” *Small Satell. Conf.*, Aug. 2021, [Online]. Available: https://digitalcommons.usu.edu/smallsat/2021/all2021/185

[97] A. Freeman, “Exploring our solar system with CubeSats and SmallSats: the dawn of a new era,” *CEAS Space J.*, vol. 12, no. 4, pp. 491–502, Dec. 2020, doi: 10.1007/s12567-020-00298-5.

[98] E. Rapuano *et al.*, “An FPGA-Based Hardware Accelerator for CNNs Inference on Board Satellites: Benchmarking with Myriad 2-Based Solution for the CloudScout Case Study,” *Remote Sens.*, vol. 13, no. 8, p. 1518, 2021.

[99] T. Speers *et al.*, “0.25 µm FLASH Memory Based FPGA for Space Applications.”

[100] T. S. Ajani, A. L. Imoize, and A. A. Atayero, “An Overview of Machine Learning within Embedded and Mobile Devices–Optimizations and Applications,” *Sensors*, vol. 21, no. 13, p. 4412, Jun. 2021, doi: 10.3390/s21134412.

[101] S. Mittal, “A survey of FPGA-based accelerators for convolutional neural networks,” *Neural Comput. Appl.*, vol. 32, no. 4, pp. 1109–1139, Feb. 2020, doi: 10.1007/s00521-018-3761-1.

[102] H. Qin, R. Gong, X. Liu, X. Bai, J. Song, and N. Sebe, “Binary neural networks: A survey,” *Pattern Recognit.*, vol. 105, p. 107281, 2020, doi: https://doi.org/10.1016/j.patcog.2020.107281.

[103] A. Munshi, “The OpenCL specification,” in *2009 IEEE Hot Chips 21 Symposium (HCS)*, 2009, pp. 1–314. doi: 10.1109/HOTCHIPS.2009.7478342.

[104] K. Guo, S. Zeng, J. Yu, Y. Wang, and H. Yang, “A Survey of FPGA-Based Neural Network Accelerator,” *ArXiv171208934 Cs*, Dec. 2018, Accessed: Nov. 15, 2021. [Online]. Available: http://arxiv.org/abs/1712.08934

[105] X. Zhao, X. Zhang, F. Yang, P. Xu, W. Li, and F. Chen, “Research on Machine Learning Optimization Algorithm of CNN for FPGA Architecture,” *J. Phys. Conf. Ser.*, vol. 2006, no. 1, p. 012012, Aug. 2021, doi: 10.1088/1742-6596/2006/1/012012.

[106] Y. Ma, Y. Cao, S. Vrudhula, and J. Seo, “An automatic RTL compiler for high-throughput FPGA implementation of diverse deep convolutional neural networks,” in *2017 27th International Conference on Field Programmable Logic and Applications (FPL)*, 2017, pp. 1–8. doi: 10.23919/FPL.2017.8056824.

[107] H. Sharma *et al.*, “From high-level deep neural models to FPGAs,” in *2016 49th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*, 2016, pp. 1–12. doi: 10.1109/MICRO.2016.7783720.

[108] R. DiCecco, G. Lacey, J. Vasiljevic, P. Chow, G. Taylor, and S. Areibi, “Caffeinated FPGAs: FPGA framework For Convolutional Neural Networks,” in *2016 International Conference on Field-Programmable Technology (FPT)*, 2016, pp. 265–268. doi: 10.1109/FPT.2016.7929549.

[109] T. Hricik, D. Bader, and O. Green, “Using RAPIDS AI to Accelerate Graph Data Science Workflows,” in *2020 IEEE High Performance Extreme Computing Conference (HPEC)*, 2020, pp. 1–4. doi: 10.1109/HPEC43674.2020.9286224.

[110] M. A. Belonosov, C. Kostov, G. V. Reshetova, S. A. Soloviev, and V. A. Tcheverda, “Parallel Numerical Simulation of Seismic Waves Propagation with Intel Math Kernel Library,” in *Applied Parallel and Scientific Computing*, Berlin, Heidelberg, 2013, pp. 153–167. doi: 10.1007/978-3-642-36803-5_11.

[111] J. Gong *et al.*, “Highly Efficient 8-Bit Low Precision Inference of Convolutional Neural Networks with IntelCaffe,” New York, NY, USA, 2018. doi: 10.1145/3229762.3229763.

[112] “Distributed FlatSats Phase 1: Use Case Scoping and Infrastructure Feasibility Study,” *SmartSat CRC*. Accessed: Nov. 16, 2021. [Online]. Available: https://smartsatcrc.com/projects/advanced-satellite-systems-sensors-and-intelligence/distributed-flatsats-phase-1-use-case-scoping-and-infrastructure-feasibility-study/

[113] D. Nunez, M. Poizat, J. Jimenez, E. Munoz, and M. Dominguez, “Enhanced Low Dose Rate Sensitivity Analysis,” in *2014 IEEE Radiation Effects Data Workshop (REDW)*, 2014, pp. 1–6. doi: 10.1109/REDW.2014.7004568.

[114] K. D. Patel and S. Belani, “Image encryption using different techniques: A review,” *Int. J. Emerg. Technol. Adv. Eng.*, vol. 1, no. 1, pp. 30–34, 2011.

[115] Y. Bentoutou, E.-H. Bensikaddour, N. Taleb, and N. Bounoua, “An improved image encryption algorithm for satellite applications,” *Adv. Space Res.*, vol. 66, no. 1, pp. 176–192, 2020.

[116] I. Kostenko, “Current Problems and Challenges in International Space Law: Legal Aspects,” *Adv. Space Law*, vol. 5, no. 1, 2020.

[117] E. Palmerini, A. Bertolini, F. Battaglia, B.-J. Koops, A. Carnevale, and P. Salvini, “RoboLaw: Towards a European framework for robotics regulation,” *Robot. Auton. Syst.*, vol. 86, pp. 78–85, 2016.

[118] L. Soroka and K. Kurkova, “Artificial Intelligence and Space Technologies: Legal, Ethical and Technological Issues,” *Adv. Space Law*, vol. 3, May 2019, doi: 10.29202/asl/2019/3/11.

USER INFORMED INDUSTRY DRIVEN RESEARCH POWERED

SMARTSAT
COOPERATIVE RESEARCH CENTRE

Building Australia's Space Industry

Australian Government
Department of Industry, Science, Energy and Resources

AusIndustry
Cooperative Research Centres Program

SmartSat CRC Head Office:
Lot Fourteen, Level 3, McEwin Building
North Terrace, Adelaide, SA

info@smartsatcrc.com
smartsatcrc.com