TALLINN UNIVERSITY OF TECHNOLOGY
School of Information Technologies

Aleksandr Bosler 232075IACM

# Adaptive Energy-Efficient CNN for Onboard Wildfire Detection on CubeSats

Master's thesis

Supervisor: Aleksei Tepljakov
PhD
Co-Supervisor: Uljana Reinsalu
PhD

Tallinn 2025

TALLINNA TEHNIKAÜLIKOOL
Infotehnoloogia teaduskond

Aleksandr Bosler 232075IACM

# Adaptiivne ja energiatõhus konvolutsiooniline närvivõrk metsapõlengute tuvastamiseks CubeSatidel

Magistritöö

Juhendaja: Aleksei Tepljakov
PhD
Kaasjuhendaja: Uljana Reinsalu
PhD

Tallinn 2025

# Author’s declaration of originality

I hereby certify that I am the sole author of this thesis. All the used materials, references to the literature and the work of others have been referred to. This thesis has not been presented for examination anywhere else.

Author: Aleksandr Bosler

12.05.2025

# Abstract

This thesis presents an adaptive, energy-aware convolutional neural network (CNN) for on-board wildfire detection in 6U CubeSats. The model augments a lightweight MobileNetV2 backbone with two confidence-driven early-exit classifiers that terminate inference once prediction certainty passes configurable thresholds. Trained on 6000 multispectral Sentinel-2 patches (3000 fire / 3000 non-fire) gathered over California, Australia and the Amazon, the network attains 97.0% overall accuracy and 96.9% recall on a held-out test set.

When deployed on a 5 W NVIDIA Jetson Nano, the adaptive configuration lowers mean per-image latency from 70.4 ms to 46.1 ms and cuts energy from 124 mJ to 70 mJ, delivering 34% faster and 44% more efficient inference while preserving baseline performance. Early exits resolve 80% of non-event images in shallow layers, reducing average compute by 30%. Scene-level analysis confirms that a full 109 × 109 km Sentinel-2 swath can be processed in 111 s at an average 1.5 W, permitting up to seven scenes per orbit within typical CubeSat power budgets.

These results demonstrate that conditional inference enables timely, high-fidelity wildfire alerting from orbit without violating strict power and thermal constraints, and they offer a transferable template for other low-incidence Earth-observation tasks requiring autonomous, energy-efficient edge AI.

This thesis is written in English and is 78 pages long, including 7 chapters, 11 figures and 11 tables.

# Annotatsioon

# Adaptiivne ja energiatõhus konvolutsiooniline närvivõrk metsapõlengute tuvastamiseks CubeSatidel

Käesolev magistritöö esitleb adaptiivset, energiateadlikku konvolutsioonilist närvivõrku (CNN) pardal toimuva metsapõlengute avastamise jaoks 6U CubeSat-idel. Mudel täiendab kerget MobileNetV2 selgroogu kahe kindlustundel põhineva varajase väljumise klassifitseerijaga, mis lõpetavad järeldusprotsessi niipea, kui ennustuse usaldusväärsus ületab määratavad läved. California, Austraalia ja Amazoni kohal kogutud 6000 mitmespektrilise Sentinel-2 lõigu (3000 põlengu ja 3000 mittepõlengu) põhjal treenitud võrk saavutas hoidunud testkogumil 97,0% üldise täpsuse ja 96,9% tagasikutsutavuse.

Platvormil 5 W NVIDIA Jetson Nano vähendas adaptiivne konfiguratsioon keskmist pildipõhist latentsust 70,4 ms-lt 46,1 ms-ni ja energiakulu 124 mJ-lt 70 mJ-ni, pakkudes 34% kiiremat ja 44% energiatõhusamat järeldust, säilitades samas lähtetaseme täpsuse. Varajased väljumised lahendavad 80% mittesündmuse piltidest madalates kihtides, vähendades keskmist arvutusmahtu 30%. Stseeni taseme analüüs kinnitab, et tervet 109 × 109 km Sentinel-2 ribalaiust saab töödelda 111 sekundiga keskmisel 1,5 W võimsusel, mis võimaldab tüüpilise CubeSat-i energiaeelarve piires kuni seitset stseeni ühe orbiidi kohta.

Tulemused näitavad, et tingimuslik järeldus võimaldab õigeaegset ja kõrgekvaliteedilist metsapõlengute teavitamist orbiidilt, rikkumata rangeid energia- ja termopiiranguid, ning pakuvad ülekantavat raamistikku teisteks madala esinemissagedusega Maa-vaatluse ülesanneteks, mis vajavad autonoomset ning energiasäästlikku edge-tehisintellekti.

Lõputöö on kirjutatud Inglise keeles ning sisaldab teksti 78 leheküljel, 7 peatükki, 11 joonist, 11 tabelit.

# List of abbreviations and terms

| | |
|---|---|
| ABI | Advanced Baseline Imager |
| AI | Artificial Intelligence |
| ASIC | Application-Specific Integrated Circuit |
| CNN | Convolutional Neural Network |
| COTS | Commercial Off-The-Shelf |
| CPU | Central Processing Unit |
| CUDA | Compute Unified Device Architecture |
| EPS | Electrical Power System |
| ESA | European Space Agency |
| FIRMS | Fire Information for Resource Management System |
| FN | False Negative |
| FP | False Positive |
| FPGA | Field-Programmable Gate Array |
| FRP | Fire Radiative Power |
| FTA | Fire Thermal Anomaly |
| GEE | Google Earth Engine |
| GEO | Geostationary Earth Orbit |
| GOES | Geostationary Operational Environmental Satellite |
| GPU | Graphics Processing Unit |
| IR | Infrared |
| LEO | Low Earth Orbit |
| MODIS | Moderate Resolution Imaging Spectroradiometer |
| NIR | Near-Infrared |
| SWIR | Short-Wave Infrared |
| TOPS | Tera Operations Per Second |
| TPU | Tensor Processing Unit |
| VIIRS | Visible Infrared Imaging Radiometer Suite |
| VPU | Vision Processing Unit |

# Table of contents

# List of figures

# List of tables

# 1 Introduction

Over the past few decades, space technology has transformed from a niche scientific endeavor into a critical infrastructure underpinning modern society. The orbital satellite network–comprising government, commercial, and scientific satellites–now plays an essential role in global communications, navigation, weather forecasting, and Earth observation. The evolution of this network is striking: while early satellite launches numbered only in the hundreds per year–for example, in 2014 there were only 241 objects launched into space–recent figures reveal an exponential increase in space activity. In 2024 alone, more than 2,800 objects were launched into orbit [1], as shown in Figure 1, and current estimates suggest that there are over 11,000 active satellites worldwide [2]. This dramatic growth is largely driven by the advent of small satellites and mega-constellations–like SpaceX's Starlink–which have redefined what is possible in terms of global broadband connectivity and remote sensing. Moreover, it is forecasted that if current trends continue, the number of active satellites could reach around 27,000 by the end of 2030, underscoring the rapid expansion of our orbital infrastructure [3].

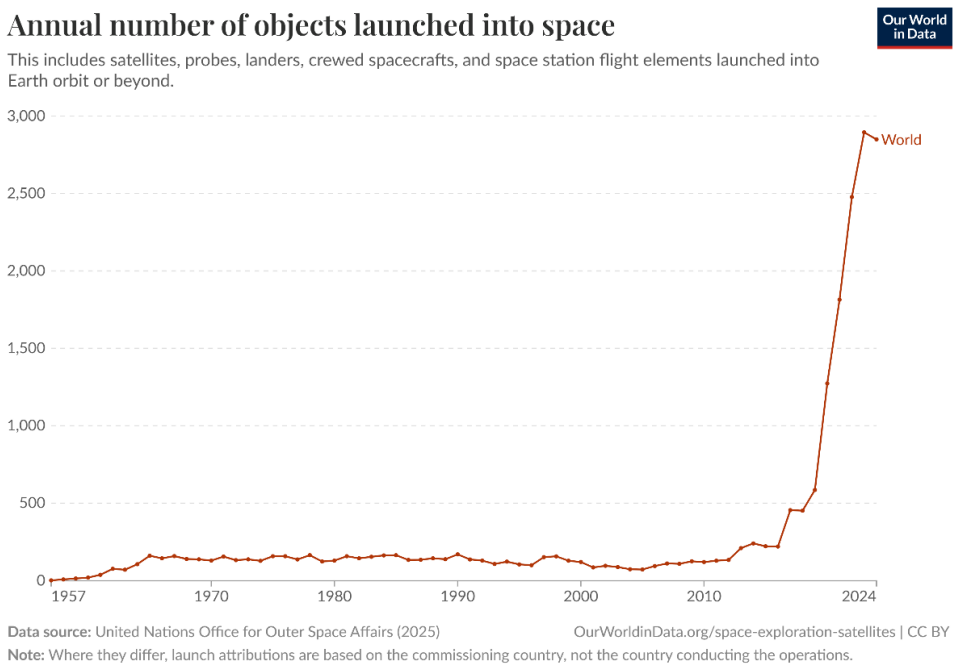



Figure 1. Annual Number of Objects Launched into Space Worldwide [1]

Such a rapid expansion of the orbital network not only demonstrates the increasing demand for space-based services but also underscores the technical challenges that accompany these advances. As satellite numbers surge, the need for innovations in onboard processing and energy management becomes paramount, especially for resource-constrained platforms such as CubeSats. Optimizing energy efficiency in these satellites is crucial to prolonging their operational life and ensuring the reliability of the services they provide. In this context, improving the energy efficiency of onboard systems–particularly those handling data-intensive tasks like image processing–emerges as a key enabler for sustaining and further expanding our space capabilities.

Small satellites are increasingly being equipped with onboard image processing capabilities to perform autonomous tasks in space, but they operate under severe energy constraints. CubeSats in the 3U–12U class (1–15 kg mass) can typically harvest only a few watts of power (on the order of 1–7 W) from their limited solar panel area [4]. Energy is stored in small batteries (often only tens of Wh), which must power all subsystems during eclipse periods. As a result, the power supply is a limiting factor for the satellite's lifecycle [5]. It is extremely difficult to carry enough stored energy for long-duration operations, so efficient power usage is critical. Every onboard operation–communication, attitude control, and especially computation–draws from a very limited energy budget. Introducing energy-intensive computing payloads can deepen battery discharge cycles and shorten the satellite's operational life. In other words, increasing onboard processing without proper optimization risks faster battery depletion and a reduced mission lifespan. Therefore, optimizing computational processes for energy efficiency is crucial to extending satellite mission duration.

One prominent example of managing this balance is the recent demonstration of artificial intelligence on a CubeSat. In 2020, the Φ-Sat-1 mission (a 6U CubeSat) carried an Intel Movidius Myriad 2 vision processing unit to perform onboard image analysis for cloud detection [6]. This was the first use of a deep neural network in orbit and it was motivated by a practical power and bandwidth problem: about two-thirds of Earth imagery is obscured by clouds, and transmitting those images wastes valuable downlink bandwidth and energy. By using an efficient onboard CNN (Convolutional Neural Network) to automatically discard cloudy images, the satellite saved roughly 30% of downlink bandwidth (and the associated transmission energy). The key to this success was the use of specialized low-power hardware–the Myriad 2 VPU (Vision Processing Unit)–which

delivers high compute performance in a few watts. This hardware was specifically designed for “impressive compute capability but in a very low power envelope”, making it well-suited for space applications. Φ-Sat-1 proved that with the right optimizations, onboard image processing can drastically reduce data handling costs without exhausting the power budget.

These trends underscore the need for energy-saving methods in onboard image analysis for CubeSats. Reducing the computational load of onboard tasks directly decreases power consumption and thermal stress on the satellite. This not only prevents brown-outs and data bottlenecks, but can also extend the effective lifespan of the platform by avoiding deep battery discharge cycles. Recent studies on satellite edge computing confirm that smart energy management (scheduling and optimizing onboard computations) can significantly prolong battery life–one optimization framework showed a more than 30% extension in battery longevity through careful task scheduling [4]. In summary, the relevance of this research lies in addressing the power limitations of small satellites: optimizing an onboard CNN for energy efficiency is pursued to enable advanced missions (such as real-time wildfire monitoring) within the tight power budgets of a 6U CubeSat.

Wildfires have become a global crisis in recent years, growing in frequency and intensity across many regions. Climate change and land use patterns have lengthened fire seasons and increased the risk of large, uncontrollable fires. For example, in the United States, an average of 70,000 wildfires occur annually, burning millions of acres of land and destroying homes, infrastructure, and lives [7]. The first half of 2022 alone saw over 50 major wildfire events in the U.S., exceeding the 10-year average, with over 190,000 acres burned by mid-year. The impacts are not only local–the ESA (European Space Agency) estimates around four million square kilometers of land are affected by wildfires each year [5]. These fires release vast amounts of carbon emissions and pollutants, harming air quality and contributing to climate change. The economic and ecological costs of such wildfires can be catastrophic, making improved wildfire management an urgent priority.

Early detection of wildfires is widely recognized as one of the most crucial factors in mitigating their damage. If a fire can be detected in its nascent phase–when it is still small and localized–there is a far greater chance that firefighters or automated suppression systems can contain it before it spreads. Studies have shown that the potential devastation of a wildfire can be minimized if it is detected and precisely located at an early stage [8].

In practical terms, early ignition detection greatly increases the likelihood of timely containment, which in turn saves lives and reduces property losses. According to the U.S. Department of Homeland Security, catching a wildfire shortly after ignition can mean the difference between a minor incident and a disaster, as it enables first responders to mobilize before the fire grows large [7]. Every minute counts: a delay in detection allows a fire to expand exponentially, complicating firefighting efforts.

Satellites and aerial sensors play a pivotal role in rapid wildfire detection. Geostationary satellites like Himawari-8 and GOES (Geostationary Operational Environmental Satellite) now monitor Earth with updates every 5–10 minutes, providing near-real-time observation that is conducive to catching fires shortly after they ignite [9]. However, traditional satellite fire detection approaches typically involve downloading large volumes of imagery to ground stations for processing, which introduces latency [10]. A CubeSat constellation with onboard fire detection capabilities could significantly speed up this pipeline–detecting fires on-board and sending immediate alerts down to authorities, rather than waiting for ground processing. By minimizing the delay between image capture and fire identification, such systems help ensure that fires are attacked in their infancy, when they are most manageable. Early wildfire detection is not just about technology but about buying precious time in emergency response. This research focuses on that critical early window, aiming to leverage advanced CNN techniques onboard a satellite to automatically detect wildfires in images within seconds of capture. In doing so, it addresses a key societal need: mitigating the impact of wildfires through faster technological intervention.

The main objective of this research is the development of an adaptive, energy-efficient CNN for onboard wildfire detection. In particular, a multi-output CNN architecture is targeted, capable of adapting its computation based on the input image content to provide rapid wildfire identification while minimizing energy consumption on the satellite.

Based on this rationale, the central hypothesis of the research is formulated as follow: *Most images that do not contain a wildfire can be identified with high confidence after only a few early layers of the CNN, allowing the system to exit early and skip the remaining processing. By filtering out clear non-fire cases in this manner, the network can reserve full computational effort for the smaller subset of ambiguous or fire-containing images. This selective allocation of resources is expected to significantly*

*reduce average processing time and energy consumption per image, without substantially compromising detection accuracy.*

Prior work on early-exit neural networks supports this idea, showing that a large proportion of inputs can be confidently classified at intermediate layers, while only the more challenging cases require full network depth [11]. This research aims to exploit this behavior in the context of wildfire imagery to achieve high detection accuracy (targeting at least 90% correct fire identification) while substantially reducing unnecessary computations for the majority of non-fire images.

To achieve this objective, the following specific research tasks are defined:

1. Identify and customize a lightweight CNN architecture that serves as the backbone for wildfire image analysis. By comparing candidates in terms of accuracy, model size, and computational cost, an optimal baseline network is selected that offers a strong accuracy-efficiency trade-off for onboard inference.
2. Integrate early-exit capability into the CNN by adding intermediate classifiers at selected depths, along with a confidence evaluation module. During inference, if an intermediate classifier is sufficiently confident, the model outputs a result immediately–skipping deeper layers. The key challenge is to determine confidence thresholds that balance energy savings with reliable wildfire detection.
3. Integrate and deploy the adaptive CNN on a hardware, which will serve as the onboard inference accelerator. Integration testing will ensure that the adaptive behavior (early exits) functions correctly on hardware and that the system meets timing constraints.

The structure of this thesis is organized as follows. The first chapter provides an overview of existing wildfire detection methods and onboard hardware platforms for satellite data processing. The second chapter describes the data preparation pipeline, including spectral band selection and augmentation techniques. The third chapter focuses on the architecture of the adaptive convolutional neural network and the training strategy. The fourth chapter presents experimental results, including an analysis of detection accuracy and energy consumption. The fifth chapter summarizes the key findings and discusses directions for future research.

# 2 Literature Review

Wildfire detection from satellite data is a rapidly evolving field, featuring methods that range from classical threshold-based techniques to advanced AI (artificial intelligence) algorithms. This chapter reviews the main approaches, highlighting both established practices and the latest innovations. Section 2.1 covers modern fire detection techniques, including threshold-based methods and the use of small satellite constellations for rapid response. Section 2.2 focuses on neural network applications for onboard processing, while Section 2.3 delves into deep learning strategies for event detection. Subsequently, Section 2.4 explore power and resource constraints on CubeSats, which directly impact the feasibility of AI-based wildfire monitoring. Finally, Section 2.5 presents hardware accelerators that enable real-time processing on resource-limited satellite platforms.

## 2.1 Modern Methods for Fire Detection

Wildfire detection from satellite imagery has evolved considerably over the past decades. While classical threshold-based methods remain widely used, more advanced techniques involving constellations of small satellites and onboard artificial intelligence (AI) are gaining traction. This section reviews both the traditional approaches and the emerging trends in fire detection from space.

### 2.1.1 Classical Threshold-Based Approaches

Satellite-based wildfire detection has traditionally relied on classical remote sensing techniques rather than deep learning. The most widely used algorithms for active wildfire detection from space are based on detecting thermal anomalies in IR (Infrared) imagery [12]. Wildfires, even relatively small ones, exhibit very high temperatures (often above 600 K) at the fire front, especially in the mid-wave infrared (~3.5–4.65 μm) band [13]. Satellite sensors such as MODIS (Moderate Resolution Imaging Spectroradiometer), VIIRS (Visible Infrared Imaging Radiometer Suite), GOES, and Himawari have IR channels that can capture this signal. The classical approach (pioneered with MODIS's MOD14 algorithm) works roughly as follows: For each pixel, the brightness temperature

at 4 μm (T4) is compared to a fixed threshold (e.g., ~310 K for MODIS daytime) and also to the background temperature of neighboring pixels [14]. A pixel is flagged as "fire" if it is significantly hotter than its surroundings and exceeds certain absolute temperature thresholds [13]. Multiple contextual tests are applied to avoid false alarms–for instance, checking that the 4 μm signal is high relative to the long-wave IR (11 μm) signal (since fires have a stronger contrast at 4 μm [15]). Additional criteria may include: the pixel's 11 μm temperature being above ambient, the difference T4–T11 exceeding a threshold, and consistency checks across successive observations [14].

Various combustion indices or decision rules have been formulated. One example is the FRP (Fire Radiative Power) which quantifies the fire intensity from the mid-IR signal. Simpler indices use band ratios or differences (like the difference between 4 μm and 11 μm brightness temperatures) to distinguish fire pixels [12]. These classical algorithms are essentially a set of threshold conditions derived empirically for each sensor. For instance, Himawari-8's wildfire detection method uses the normalized deviation of the 3.9 μm brightness from the background to identify potential fire pixels [16]. The advantage of threshold-based methods is that they are computationally light (simple arithmetic comparisons) and have been honed over decades to be reasonably reliable. They are currently operational: NASA's MODIS Active Fire product and NOAA's GOES wildfire alerts are built on such rules.

However, there are significant limitations to these classical approaches. First, the fixed thresholds struggle with varying conditions–e.g., different surface temperatures, solar reflection at 4 μm in daytime, or sensor noise. To maintain high confidence, the thresholds are usually set conservatively, which leads to missed detections (omissions) of cooler or smaller fires. In the early stages of a wildfire (or for fires under clouds), the thermal signature might be subtle and thus go undetected by a rigid threshold test. Second, false positives can occur due to other hot surfaces (sun-heated rocks, industrial heat sources) or reflection of sunlight in the IR bands (particularly at dawn/dusk for geostationary sensors) [14]. The contextual tests mitigate this, but some false alarms still slip through, requiring human analyst confirmation. Another issue is the coarse spatial resolution of many operational sensors–for example, GOES-16's ABI (Advanced Baseline Imager) has ~2 km resolution at sub-satellite point for the 3.9 μm band. A fire must be fairly large (or very intense) to noticeably raise the temperature of a 2 km pixel [17]. This means small or early fires often are undetectable until they grow. As a result, threshold methods on

current satellites typically detect only moderate-to-large wildfire events, and there can be a delay until the fire grows enough to be picked up [18].

Comparative studies have further demonstrated the limitations of these threshold-based methods when applied to orbital wildfire detection. For instance, FireCNN, a deep learning model trained on imagery from the Himawari-8 geostationary satellite, achieved a 35.2% improvement in detection accuracy compared to conventional threshold algorithms. In addition to enhanced accuracy, FireCNN was capable of producing predictions across large regional satellite scenes in under 4 seconds, enabling near-real-time fire alerting [19]. These results highlight the growing performance gap between traditional rule-based methods and CNN-driven approaches in the context of satellite-based fire monitoring, especially for detecting early-stage or small-scale ignition events.

### 2.1.2 Satellite Systems in GEO, Polar Orbits and CubeSat Constellations

Geostationary satellites like NOAA's GOES-16/17 and JMA's Himawari-8 have become important for wildfire monitoring due to their high temporal frequency (updates every 5–10 minutes). GOES-16's ABI, for instance, continually scans the Americas and can spot fires much faster than polar-orbiting satellites (which might only pass twice a day). GOES uses a FTA (Fire Thermal Anomaly) algorithm (an evolution of the Wildfire Automated Biomass Burning Algorithm, WFABBA). This algorithm applies dynamic thresholds taking into account satellite viewing angle and uses multiple IR bands. A recent improvement in the GOES-16 FTA algorithm increased active fire pixel detection by ~6% compared to the previous GOES generation. By tailoring the thresholds to the new sensor's radiometric performance and incorporating refined contextual tests, GOES-16 was able to detect more fires–especially smaller or cooler ones–that had been missed by GOES-13 [20]. Nevertheless, even with these improvements, GOES fire detection still faces the fundamental challenge of resolution. Himawari-8, covering the Asia-Pacific region, similarly employs threshold and contextual methods. Studies have shown Himawari's algorithm can detect large fire events but might omit fires below a certain size or in heterogeneous terrain due to the strict thresholds. Low spatial resolution makes identification by conventional threshold processing difficult, particularly in the case of early-stage fires or small forest fires [21]. In essence, the temporal advantage of GEO (Geostationary Earth Orbit) satellites is partially offset by spatial limitations – they can tell when a big fire starts within minutes, but they might not see a small fire at all.

Polar-orbiting satellites carrying instruments like NASA's VIIRS (375 m resolution IR) [18] and MODIS (1 km IR) have finer spatial detail and can detect smaller fires, but they only overpass a given location a few times per day. This can introduce delays of several hours in detection (fires can ignite and spread in between passes). In practice, a combination is used: polar satellites for detailed mapping and small fire detection, and GEO satellites for continuous monitoring of large fire growth [22]. Even so, current systems often miss the earliest ignition phase. For example, analysis of Australian bushfires showed that some fires were already a few hundred hectares in size by the time satellites flagged them. Additionally, heavy smoke or clouds can obscure the thermal signature, leading to missed detections [23].

Recognizing the gaps in current systems, there is growing interest in deploying dedicated small satellite constellations for wildfire detection. These promise higher spatial resolution and more frequent revisits by using many satellites in LEO (Low Earth Orbit). One prominent initiative is by the company OroraTech. OroraTech has been launching a series of CubeSats equipped with thermal infrared cameras specifically tuned for fire detection. In 2022, OroraTech launched its first demo CubeSat [24], and by March 2025 they had launched 8 wildfire detection satellites on a Rocket Lab Electron rocket [25]. This is Phase 1 of a planned constellation of up to 100 satellites by 2028, aiming to provide 24/7 global coverage with a target revisit of ~30 minutes [26]. Each OroraTech satellite carries an uncooled microbolometer thermal imager and on-board processing to detect fire hotspots in real time. The data from these satellites feed into OroraTech's wildfire monitoring platform which fuses them with existing satellite data. Early reports indicate the OroraTech system can detect fires as small as a few tens of meters across under ideal conditions, and provide alerts to users with low latency [25]. The on-board software likely uses a mix of thresholding and machine learning to distinguish real fires from false signals. OroraTech's approach exemplifies the new wave of private-sector missions tackling wildfires with constellations of CubeSats.

Another notable project is FireSat, backed by the Earth Fire Alliance (a non-profit) and partners like Muon Space and Google. FireSat is envisioned as a constellation of about 50 fire-focused satellites in LEO, each with high-resolution thermal sensors, to achieve "neighborhood-scale" fire monitoring globally. The goal is to detect fires on the order of 5×5 meters in size and deliver alerts within minutes of observation [27]. The system has entered its initial deployment phase, with the first satellite launched in March 2025 [28]

and several more planned as part of the first operational cluster in 2026. FireSat plans to leverage modern onboard processing and high-resolution thermal imaging to deliver near real-time fire perimeter and radiative power maps, providing actionable information to first responders and incident managers–a strong use-case for edge AI in orbital fire monitoring [27].

## 2.2 Neural Network Implementations for Onboard Processing

In recent years, there have been pioneering demonstrations of running neural networks directly onboard satellites, marking a shift from the traditional paradigm of sending raw data to the ground for processing. These early missions illustrate the feasibility and benefits of AI at the edge in space. This section reviews several notable examples–including the European Space Agency's Φ-Sat missions and other smallsat projects–focusing on their onboard hardware, use cases, and outcomes.

### 2.2.1 ESA's Φ-Sat Missions

Φ-Sat-1 (2020). As previously noted in the Introduction, ESA's Φ-Sat-1 was a 6U CubeSat developed under the FSSCat program, carrying an onboard AI system for cloud detection. It featured an Intel Movidius Myriad 2 VPU–integrated by Ubotica Technologies–to classify and discard cloud-contaminated images before downlink, conserving bandwidth. The onboard CNN ("CloudScout") ran successfully in-flight, filtering hyperspectral image tiles and transmitting only useful data. Results confirmed both the reliability of inference and the Myriad 2's radiation resilience, with no significant upsets reported. The mission demonstrated that onboard decision-making can improve bandwidth use and data latency, paving the way for more advanced AI payloads in space [6].

Φ-Sat-2 (2024)–Building on the success of Φ-Sat-1, the Φ-Sat-2 mission was initiated to further explore on-board AI for Earth observation. Φ-Sat-2 is a 6U CubeSat developed by a consortium led by Open Cosmos, with a goal of running a variety of AI applications on orbit. The satellite carries a powerful onboard computer with an AI accelerator–specifically, the Intel Movidius Myriad 2–and a multispectral camera with a resolution of 4.75 m [29]. Uniquely, Φ-Sat-2 was designed as a software-defined platform capable of hosting multiple AI apps that can be uploaded and updated during flight. The satellite currently runs several AI applications, namely SAT2MAP (which extracts street map data

for emergency scenarios), Autonomous Vessel Awareness (AVA) (which autonomously detects and classifies vessels in optical imagery), Cloud Detection (an expanded system from Φ-Sat-1), and Deep Compression (which compresses images onboard using a convolutional autoencoder). Additionally, two further applications were uploaded post-launch from the OrbitalAI challenge (an international competition promoting innovative onboard AI solutions): one detects anomalies in marine ecosystems, and PhiFire AI is designed for wildfire detection [30]. Essentially, Φ-Sat-2 serves as an in-orbit testbed for edge computing, where algorithms can be deployed and evaluated in space. It stands as one of the most advanced demonstrations of reconfigurable AI in space–showing how a single CubeSat can flexibly run different neural networks for different missions. While results from Φ-Sat-2 are still forthcoming, the mission underlines a trend: public-private partnerships (ESA with startups like Ubotica, Open Cosmos) are pushing the state-of-the-art in onboard AI, moving beyond fixed-function inference to a more dynamic "app store" model for satellites.

### 2.2.2 Other Onboard AI Demonstrations

KITSUNE 6U Wildfire CubeSat (2022)–A domain-specific example of onboard AI for wildfire detection is provided by the KITSUNE mission. Developed by the Kyushu Institute of Technology in collaboration with partners, this 6U CubeSat was deployed from the ISS in 2022 with a 5-meter resolution imager. Its mission includes demonstrating a deep learning approach for classifying images containing wildfires directly onboard the satellite. KITSUNE is the first CubeSat to employ a CNN to classify wildfire images in LEO, with the aim of reducing downlink data by performing image processing in orbit. The team developed a lightweight CNN (Mini-VGGNet-based) and pre-trained it on a dataset of wildfire and non-wildfire images. This model was designed to run on KITSUNE's onboard computer as a secondary mission, analyzing captured images for wildfire smoke or burn signatures. If a wildland fire is detected with high confidence, the satellite can prioritize downloading those images (or even just alerts) to the ground, rather than streaming all raw imagery. This design addresses the bandwidth bottleneck by transmitting only meaningful data (similar in spirit to Φ-Sat's cloud filtering). While KITSUNE's primary imaging is in visible bands (RGB) and thus limited to daytime fire or smoke detection, its CNN approach has shown high accuracy in laboratory testing. The Mini-VGGNet-based CNN has demonstrated 98% accuracy with 0% false positives while consuming only 0.10 Wh of energy under power-cycled conditions without hardware

acceleration [31]. Although the mission ended in March 2023 when the satellite re-entered Earth's atmosphere [32], it provided a valuable demonstration of onboard wildfire classification. This project underscores how even university-built CubeSats are now leveraging CNNs in orbit, validating their applicability for environmental monitoring.

While KITSUNE targeted wildfire detection specifically, other CubeSat and small-satellite initiatives have turned to AI to address a wide range of Earth observation challenges. For example, D-Orbit's ION satellite platform recently hosted an experiment called WorldFloods (2022) where a deep neural network segmented flood maps from imagery entirely onboard, using an Intel Myriad X accelerator. The WorldFloods payload produced 10 m resolution flood masks in near real-time and downlinked vector map outputs instead of raw images, drastically cutting response time for flood events [33]. Another example is Lockheed Martin's 2020 announcement of a 3U CubeSat carrying an NVIDIA Jetson GPU (Project La Jument) to test AI processing in orbit. The GPU-enabled satellite was intended to run applications like image super-resolution and object detection on-board [34]. Although challenges like radiation hardening of COTS (Commercial Off-The-Shelf) GPUs remain, these initiatives show that even more computationally intensive AI (beyond CNN classification) is being trialed in space.

## 2.3 Deep Learning Methods for Event Detection

Convolutional Neural Networks (CNNs) have become the cornerstone of image-based event detection, including wildfires. Unlike earlier algorithms relying on manually crafted features, CNNs automatically learn feature hierarchies from data–from low-level edges to high-level shapes–making them highly effective for complex pattern recognition. In the domain of wildfire detection, numerous studies have successfully leveraged CNNs to identify fire or smoke in images and videos. Approaches include multi-stage strategies where candidate moving regions are first isolated using traditional image-processing techniques such as background subtraction and dark-channel priors, followed by CNN-based classification. Other techniques fuse deep learning with traditional methods by combining CNN-extracted static image features with dynamic features from optical-flow algorithms. Additionally, common practices involve transfer learning, where pretrained CNN models, such as VGG16 and ResNet50, are fine-tuned specifically for wildfire detection tasks, often emphasizing datasets rich in non-fire examples to minimize false

positives. Smaller, lightweight CNN architectures inspired by widely recognized networks like GoogleNet have also been effectively applied to flame detection in surveillance videos. Collectively, these CNN-based methods consistently outperform earlier smoke sensors and threshold-based algorithms, demonstrating superior performance in early fire detection across diverse conditions [34].

Despite their high accuracy, standard deep CNN models such as VGG or ResNet come with substantial computational and energy demands. This limitation is especially critical for resource-constrained environments like CubeSats, which have limited onboard processing power and strict energy budgets. To address this, researchers have developed lightweight and energy-efficient CNN architectures optimized for low-power devices. Notable examples include MobileNet, EfficientNet, and various TinyML approaches for microcontrollers. These models drastically reduce the number of parameters and operations, enabling inference on embedded hardware with minimal accuracy loss.

MobileNet was one of the first CNN architectures designed explicitly for mobile and embedded vision applications. It introduces depthwise separable convolutions, factorizing a standard convolution into a depthwise spatial convolution followed by a pointwise convolution. This reduces computation and model size by nearly an order of magnitude compared to conventional CNNs, with MobileNet-V1 (4.2 million parameters) achieving ~70% ImageNet top-1 accuracy–only modestly lower than much larger models [35]. Successive versions like MobileNet-V2 improved upon this by using inverted residual blocks and linear bottlenecks, further boosting efficiency. Such architectures demonstrate that with carefully designed layers, it is possible to run CNN inference within the tight resource constraints of small satellites [36]. Indeed, prior work has shown MobileNet models can run in real-time on low-power processors typical of CubeSats [37].

EfficientNet represents another class of optimized CNNs that leverage neural architecture search and compound scaling. Tan and Le (2019) proposed scaling network depth, width, and input resolution in a balanced way to maximize accuracy for a given compute budget. The EfficientNet family (B0 through B7) achieves superior accuracy–efficiency trade-offs; for example, EfficientNet-B7 reaches 84.3% ImageNet top-1 accuracy while being 8.4× smaller and 6.1× faster than previous CNNs of comparable accuracy. Even the smaller EfficientNet-B0 (approximately 5 million parameters) outperforms much larger models in accuracy per FLOP [38]. This level of efficiency is highly attractive for

CubeSat platforms, which cannot afford the memory and FLOPs of large CNNs. EfficientNet's compound scaling strategy ensures that a suitably scaled-down model can fit within on-board computing limits without sacrificing too much performance. Comparative studies confirm that EfficientNet and MobileNet variants are among the top choices for deploying CNNs on small-satellite hardware [37].

Beyond specific architectures, the field of TinyML focuses on implementing machine learning on ultra-low-power microcontroller units (MCUs). TinyML techniques include model compression, quantization to lower bit-width (e.g. 8-bit integers), and distillation to smaller networks. These methods can shrink CNN memory footprint and inference energy drastically, albeit often with some loss in accuracy. For instance, 8-bit quantization typically results in a reduction of model size and computational load by a factor of four or more, with minimal impact on accuracy for many vision models. Frameworks like TensorFlow Lite and tools like ARM CMSIS-NN enable deploying compact CNNs on MCU-class processors [39]. In the context of CubeSats, which often rely on low-power system-on-chips, TinyML approaches are highly relevant. Although onboard computers on CubeSats are typically more capable than bare-metal MCUs, applying TinyML strategies–such as model pruning and efficient layers–remains essential to ensure that CNNs can operate within the satellite's strict power and processing constraints.

### 2.3.1 Multi-Exit Neural Networks for Energy-Constrained Inference

While efficient architectures reduce the overall compute load of CNNs, another orthogonal strategy to save energy is making the inference process adaptive. A prominent approach is the use of multi-exit neural networks, also known as early-exit or anytime prediction networks. In a multi-exit CNN, several classifier "exit" points are inserted at intermediate layers of the network. Instead of always running the full deep network, the model can exit early at one of these intermediate points if it is confident in the prediction. For example, BranchyNet first introduced this concept, demonstrating that a CNN with multiple exits can significantly speed up inference by handling easy inputs in shallow exits and only processing hard inputs to the deeper layers. The key idea is a trade-off: shallower exits are faster (less computation) but usually less accurate; deeper exits are more accurate but consume more time/energy. During inference, the network evaluates the confidence (often using entropy or a threshold on the softmax output) at an exit. If the confidence is high, the result is output at that stage. Otherwise, the input continues to

propagate through deeper layers. This adaptive approach means computation is adjusted per input–simple cases don't waste energy on full network evaluation, while complex cases still get the benefit of the full model [11].

The benefits of multi-exit CNNs for energy efficiency are well documented. One approach involved implementing a multi-exit CNN on a small ESP32-CAM embedded device, and the results showed that it can filter out images with no objects of interest, skipping the rest of the network processing. In this system, the multi-exit design saved about 42.7% of energy compared to always running the full network (single-exit). This came at the cost of only a minor accuracy drop (~2.7% lower accuracy than the full model), a trade-off very favorable for energy-constrained scenarios [40]. Similarly, others have applied multi-exit networks in IoT settings to meet real-time requirements. In one industrial IoT application, an early-exit CNN was used to ensure that deadlines were met for time-sensitive tasks [41]. These studies confirm that adaptive computing can maintain high accuracy and dramatically cut down inference time and power consumption. Compared to static models (which have a fixed depth and computational cost for every input), adaptive models are much more flexible and efficient. A static CNN, no matter how simple or complex, cannot change its behavior on the fly–it treats an easy clear-sky image the same as a challenging smoky image, expending equal computation on both. In contrast, a multi-exit CNN might recognize very quickly that a clear image has "no fire" with high confidence after just a few layers and exit early, saving the effort of running the remaining layers. Over many images, especially in scenarios like continuous Earth observation where many frames have no event, the energy savings accumulate significantly.

Beyond multi-exit architectures, adaptive computations include techniques like dynamic model selection and layer skipping. Some recent research explores neural networks that can skip certain layers or blocks for particular inputs, using gating mechanisms to activate only the necessary computations. For instance, an approach may train a gating network to decide if an image needs a high-capacity path or a low-capacity path through the model, effectively adjusting the network's depth or width per input [42]. This is related to the concept of conditional computation. ATM-Net (Adaptive Termination and Multi-Precision Network) is one such strategy that not only allows early termination like multi-exit but also adjusts the precision of computations based on energy availability. In extreme energy-harvesting situations, systems have even tried dynamically lowering the

precision (e.g., using integer arithmetic when power is low) or alternating between high-accuracy and low-cost model versions [43]. Another avenue of efficiency is model compression techniques (pruning unimportant weights, quantizing weights to lower bit-depths)–while these are largely static optimizations done offline, they complement adaptive inference by making the overall network smaller and faster from the start. For example, a pruned or quantized model uses fewer operations, and if combined with early exits, the gains multiply. Indeed, one study proposed a "nonuniformly compressed" multi-exit network tailored for energy-harvesting devices, recognizing that both the network architecture and weight compression need to be co-designed to enable ultra-low-power operation [44].

## 2.4 Power Generation and Storage Limitations on CubeSats

CubeSats are standardized small satellites composed of 10×10×10 cm units (1U) stacked to form larger sizes. Figure 2 illustrates typical CubeSat form factors, including 1U, 2U, 3U, 6U, and beyond (e.g., 12U), corresponding to roughly 1-2 kg per 1U [45]. Smaller CubeSats have very limited surface area for solar cells and thus severely constrained power budgets. Typical available power scales with size: A 1U CubeSat may only generate on the order of 1–2.5 W of power, a 2U about 2–5 W [46], and a 3U roughly 7–22 W. These ranges assume body-mounted solar panels in low Earth orbit and can vary with sun exposure and panel technology. For example, the Delfi-C3 (3U CubeSat) had a guaranteed input power of only ~2.4 W for operations [47]. In some cases deployable solar panels or solar sails are used to boost input power, but even then the power is typically at most tens of watts [48].

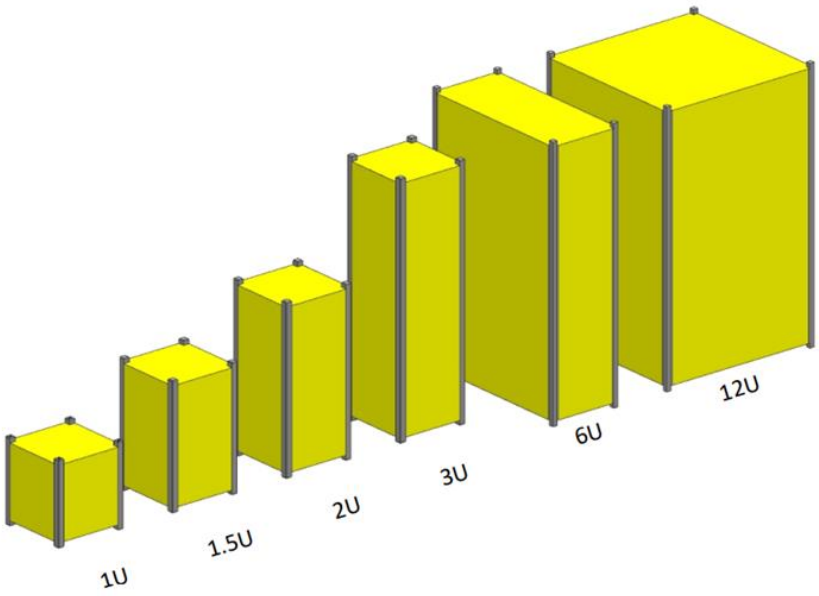



Figure 2. Common CubeSat Standard Sizes (1U to 12U) [45]

Beyond 3U, larger CubeSats like 6U and 12U can accommodate more solar cells (often with deployable arrays) and larger batteries, increasing the energy budget. A 6U CubeSat (approximately 10×20×30 cm, up to ~12 kg) can generate on the order of tens of watts under peak sunlight. Commercial 6U solar panels can produce around 17–20 W in low Earth orbit [49], and advanced deployable arrays can reach 50+ W [50]. However, practical usable power is lower once you account for orbital day/night cycles and subsystem overhead. The energy storage is also limited–CubeSat batteries (typically lithium-ion) vary in capacity from around 10 to 150 Wh, while the largest batteries, used in extended or larger CubeSat configurations, can store up to approximately 350 Wh [51]. This means available power must be budgeted carefully across all subsystems (attitude control, communications, payload, etc.) over an orbit. Thermal constraints also limit continuous high-power draw.

### 2.4.1 Energy Use for AI Workloads on 6U CubeSats

Building on the general power constraints, onboard data processing capabilities in 6U platforms are especially affected by limited energy availability and hardware throughput. Platforms such as those offered by Open Cosmos, used in missions like Φ-Sat-2, can support peak payload power levels of approximately 25 W [52]. However, it is important to note that this peak power value only represents short-term consumption spikes under favorable conditions, such as direct sunlight on solar panels and fully charged batteries.

The average available power on a CubeSat is limited by orbital lighting conditions, alternating between sunlight and eclipse. For example, the 6GStarLab mission (based on an Open Cosmos platform) specifies an average payload power between 7–20 W, with a 15% duty cycle–allowing full-power operation for about 13.5 minutes per 90-minute orbit [53]. This budget can be distributed flexibly: either in short bursts at high power (e.g., 20 W for rapid image processing) or as continuous low-power operation (e.g., 3 W sustained). For wildfire detection, CNN inference is an intermittent task that can be selectively activated during target overpasses. A common strategy is to duty-cycle the processor–briefly spiking power during active processing, then returning to idle–keeping average consumption low over the orbit.

For example, the KITSUNE satellite used a Raspberry Pi Compute Module with pre-trained CNNs that were activated only on ground command, reducing energy usage to just a few watt-hours per orbit [31]. Its imaging system consumes approximately 2.9 Wh per session, with ~0.8 Wh used during the 3-minute image acquisition phase alone [54]. This highlights the importance of activating high-power components only when necessary and managing energy carefully throughout the orbit.

## 2.5 Hardware Accelerators for AI Processing on CubeSats

To meet the stringent energy constraints while still providing adequate processing throughput, specialized hardware accelerators are used for running CNNs on small satellites. In recent years, several types of low-power AI accelerators have emerged that are suitable for CubeSat deployment: Vision Processing Units (VPUs), Tensor Processing Units (TPUs), low-power Graphics Processing Units (GPUs), and even field-programmable gate arrays (FPGAs) with custom CNN logic. Below is a review of the capabilities and limitations of prominent options.

### 2.5.1 Intel Movidius Myriad X VPU

The Intel Movidius Myriad X is a vision processing unit designed for energy-efficient deep learning inference. It builds on the earlier Myriad 2 (which was used in the first in-space AI experiments). The Myriad X features 16 SHAVE vector cores and a dedicated Neural Compute Engine for CNN acceleration. In theory, it offers computational performance exceeding 4 trillion operations per second (TOPS) for deep neural networks, while in practice it can sustain over 1 TOPS of inference throughput under typical

conditions [55]. Impressively, this performance is delivered within a tiny power envelope on the order of 1–2 W. Intel noted that Myriad X achieves ~10× the neural performance of Myriad 2 within the same sub-2W power budget [56].

For CNNs relevant to wildfire, a Myriad X can provide real-time inference capabilities while drawing only a couple watts. This VPU has been demonstrated in space: the European Φ-Sat-1 (PhiSat-1) mission, a 6U CubeSat launched in 2020, used the predecessor Myriad 2 VPU to run an onboard cloud detection AI experiment. PhiSat-1 showed that COTS VPUs can survive in LEO and perform useful filtering of images. The Myriad X, being more powerful, has since been considered for similar use. Researchers have reported that Myriad VPUs offer high performance per watt and even showed acceptable radiation tolerance for short LEO missions, though they are not radiation-hardened devices [6]. In particular, the Myriad X was tested aboard the International Space Station (ISS), where it demonstrated stable operation in a radiation-prone environment — no radiation-induced faults were observed during extended runtime [57]. The limitation of Myriad X is that it requires a host processor to feed it data and handle communications (typically it's used as a co-processor via USB or PCIe). Still, for an adaptive CNN on a CubeSat, the Myriad X provides an excellent balance of ~1 TOPS/W energy efficiency and compact size, making it a strong candidate where power is at a premium.

### 2.5.2 Google Edge TPU

The Edge TPU (Tensor Processing Unit) by Google is a small ASIC (Application-Specific Integrated Circuit) specialized for TensorFlow Lite CNN inference, well-known for its use in the Coral Dev Board and USB accelerators. It is capable of 4 TOPS peak performance (INT8) and achieves about 2 TOPS per Watt of power efficiency. In absolute terms, an Edge TPU typically operates at ~2 W for full throughput (4 TOPS) and can perform inferences with very low latency.

For example, an Edge TPU can execute mobile vision models (MobileNet, Inception, etc.) in a few milliseconds each, enabling high frame-rate analysis on just a few watts [58]. For a 6U CubeSat doing wildfire detection, the Edge TPU's appeal is its minimal power draw and simplicity of integration. It interfaces via USB or PCIe to a modest single-board computer (like a Raspberry Pi or i.MX-based OBC) and offloads the math of the neural network. Its 0.5 W per TOPS efficiency is among the best in class– meaning

it can run continuous inference with almost negligible impact on a ~10 W power budget. The Edge TPU excels at classification and object detection tasks on relatively small models. As long as the CNN is quantized and fits the Edge TPU's memory, one can expect hundreds of inferences per second for small images, or several inferences per second for megapixel images, all within a couple watts.

One consideration is that the Edge TPU (as a COTS chip) has not yet flown in space, so its behavior under radiation is based on ground testing [59]. It likely would require similar risk mitigation as other COTS parts (error-correcting memory, watchdog resets on latch-up, etc.). Overall, the Google Edge TPU offers high CNN performance for minimal energy–ideal for an energy-efficient CubeSat that needs to process images onboard in real time. It represents a commercially accessible way to achieve AI on-board, given its wide adoption in edge AI projects.

### 2.5.3 NVIDIA Jetson Series (Nano, Xavier NX, Orin Nano)

NVIDIA's Jetson line includes several low-power GPU-based modules that can run neural networks with high performance, albeit at higher power consumption than dedicated ASICs. These Jetson modules combine a GPU, CPU, and often AI accelerator cores, making them stand-alone computing solutions. Key members of the family relevant to CubeSats are discussed below.

Jetson Nano is a small module with a 128-core Maxwell GPU and quad-core ARM CPU. It delivers about 472 GFLOPS (0.472 TFLOPS) of FP16 neural compute, equivalent to roughly ~0.5 INT8 TOPS, and has a configurable TDP of 5 W or 10 W [60]. In practice, the Nano can run simpler CNN models (e.g. smaller CNNs like Mobilenet or segmentation networks) at a few frames per second. It is very low-cost and has been used in some experimental payloads [61]. However, the Nano's performance per watt (~0.05–0.1 TOPS/W) is much lower than ASIC accelerators. It can fit within a CubeSat power budget (as low as 5 W in reduced mode), but it yields limited throughput and generates significant heat for the small volume.

Jetson Xavier NX is a far more powerful module (based on NVIDIA's Volta architecture with 384 CUDA cores and 48 Tensor cores) that provides up to 21 TOPS (INT8) of AI performance at full throttle. The Xavier NX features a 6-core CPU and is available in configurations with either 8 GB or 16 GB of RAM, and can be set to 10 W or 20 W power

modes. At 20 W it can achieve its peak 21 TOPS, which is enough to run very deep CNNs (ResNet-50, YOLO detectors, etc.) in real time [62]. This level of performance on a CubeSat is cutting-edge – for example, the German OroraTech wildfire monitoring CubeSats are deploying Xavier NX onboard to analyze thermal imagery for fire detection in orbit [63]. The obvious trade-off is power and thermal load: a continuous 20 W draw is a big chunk of a 6U's budget and requires excellent thermal management (radiator surfaces or heat pipes to shed ~20 W of heat). Nonetheless, for missions where advanced onboard processing is paramount and power is available (perhaps via deployable solar panels and large batteries), the Xavier NX offers a self-contained solution to run modern CNNs at the edge.

Jetson Orin Nano, NVIDIA's latest entry (2023), pushes the performance further in a small form factor. The Orin Nano can deliver up to 67 INT8 TOPS depending on the model, with power configurable between ~7 W up to 25 W for the highest performance. It uses the Ampere GPU architecture with built-in Tensor Cores, giving it an efficiency boost over older Jetsons. In effect, an Orin Nano at 10 W may achieve tens of TOPS – an order of magnitude more than Jetson Nano for the same power. This sets a new baseline for "edge AI" capability: NVIDIA advertises up to 140× the performance of Jetson Nano when running at higher power settings [64]. For CubeSat use, the Orin Nano could be run in a throttled mode (e.g., 10 W limit) to still provide perhaps ~20+ TOPS, which might enable running multiple CNN models (e.g., object detection plus image segmentation) concurrently on-board. Again, the challenge will be power provisioning and thermal dissipation in a small satellite. But if the mission can accommodate it, Orin-based systems could perform much more complex analysis (potentially even edge training or large-scale data fusion) on the satellite.

Jetson modules provide the flexibility of a full Linux computer with integrated GPU acceleration, which is excellent for development and for the integration of complex algorithms. They also support a wide range of neural network models through NVIDIA's TensorRT and CUDA libraries, enabling the use of state-of-the-art CNN architectures. However, their energy efficiency lags behind that of dedicated VPUs/TPUs. As summarized in Table 1, for instance, while the Google Edge TPU delivers approximately 2 TOPS/W and the Intel Movidius Myriad X achieves around 0.5–2 TOPS/W, the NVIDIA Jetson Nano only offers about 0.05–0.1 TOPS/W, and even the high-performance Jetson Xavier NX reaches roughly 1 TOPS/W at a 20 W power draw. The

newer NVIDIA Jetson Orin Nano can achieve around 2 TOPS/W in energy-efficient configurations (e.g., around 10–15 W), but it also requires careful power and thermal management. Additionally, Jetson modules typically have large memory footprints and necessitate meticulous power sequencing, complicating the design of the EPS (electrical power system). In summary, while NVIDIA Jetson platforms are well-suited for high-performance onboard AI–provided that the CubeSat can continuously supply roughly 10–15 W and effectively manage heat—they might be overkill if the mission only demands a lightweight wildfire classifier.

Table 1. Comparison of Edge AI Accelerators in Terms of Performance, Power Consumption, and Energy Efficiency.

| **Accelerator** | **Performance (INT8 TOPS)** | **Power Consumption (W)** | **Energy Efficiency (TOPS/W** |
|---|---|---|---|
| Intel Movidius Myriad X | 1-4 | 1-2 | 0.5-2 |
| Google Edge TPU | 4 | ~2 | ~2 |
| NVIDIA Jetson Nano | ~0.5 | 5-10 | ~0.05–0.1 |
| NVIDIA Jetson Xavier NX | Up to 21 | 10-20 | ~1 |
| NVIDIA Jetson Orin Nano | Up to 67 | 7-25 | ~2 (Under typical settings) |

### 2.5.4 Other Accelerators and FPGA-Based Solutions

Beyond the mainstream options above, there are other specialized hardware solutions that can be considered. Field-programmable gate arrays (FPGAs) can be programmed to implement CNN inference logic with parallelism and low power, and some space-grade FPGAs are radiation-tolerant. Modern low-power FPGAs (like Xilinx Zynq or Microchip PolarFire SoC) can achieve respectable performance. In fact, researchers have demonstrated CNN acceleration on a Xilinx Zynq-7020 (Artix-7 class) FPGA (Field-Programmable Gate Array) in a 1U CubeSat context: using model quantization (to 4-bit weights) and parallelism, they achieved about 15 frames per second throughput at only ~2.5 W power consumption on a cloud detection CNN [65]. This is an excellent 0.2 W per inference efficiency, showing that FPGAs can be competitive in energy efficiency. The advantage of an FPGA design is that it can be tailored exactly to the needed CNN

(saving power by not having unused circuitry) and, if using a space-grade FPGA, can offer high immunity to radiation upsets. For example, Xilinx's XQRKU060 (Kintex UltraScale) or Microchip's RTG4 are radiation-hardened FPGAs that have enough logic to implement small-to-medium CNNs [66]. The downside is development effort: one must design and verify the CNN accelerator in HDL or use high-level synthesis, which is more complex than deploying on a GPU/VPU. Also, FPGAs generally have lower absolute performance; a single small FPGA might not reach more than a few TOPS unless it's a high-end device. Nonetheless, for ultra-energy-efficient or radiation-critical missions, FPGA accelerators are very attractive. They can also be reconfigured in-flight if needed (to update the CNN architecture, for instance) [67].

At the lower end, there are high-performance microcontrollers/DSPs (e.g., the ARM Cortex-M55 with Ethos-U NPU, or Texas Instruments C7x DSP) that can perform simpler CNN inference at milliwatt power levels [68]. These might handle tinyML models for fire detection (for instance, a very lightweight CNN to detect hotspots in low-resolution thermal data). They are highly power-efficient for small models, but likely insufficient for more robust image classification tasks without significant compromises in accuracy or speed.

# 3 Data and Preprocessing

The effectiveness of a convolutional neural network for onboard wildfire detection largely depends on the quality of the input data and the proper preparation of the dataset. This chapter describes the sources of satellite imagery, the process of transforming the data into a format suitable for training, and the preprocessing techniques applied. Particular attention is given to patch extraction, handling of multispectral channels, data calibration, and augmentation methods that enhance the model's generalization capabilities.

## 3.1 Data Sources

This study relies on a combination of satellite-based fire detection records and high-resolution multispectral imagery to construct a dataset suitable for training and evaluating an onboard wildfire detection model. The foundation of the labeling process is the FIRMS (Fire Information for Resource Management System), a service operated by NASA that provides global, near-real-time data on active fire events . Fire labels were obtained from the Visible Infrared Imaging Radiometer Suite (VIIRS) sensors onboard the Suomi NPP satellite, covering the years 2021 to 2024 [69]. VIIRS detects thermal anomalies on Earth's surface by monitoring mid- and thermal-infrared bands, making it a reliable source for identifying wildfire hotspots. It offers a spatial resolution of approximately 375 meters and provides multiple daily observations per region, enabling timely and geographically precise fire event identification. VIIRS data were used specifically to determine fire occurrence locations and timestamps, serving as reference points for identifying associated satellite images.

To obtain corresponding satellite imagery, Sentinel-2 data were used. This constellation, operated by the ESA, captures multispectral images across 13 spectral bands at spatial resolutions ranging from 10 to 60 meters. Multispectral imagery refers to data captured across several wavelength ranges, including visible light and infrared, which is particularly effective for detecting vegetation stress, heat signatures, and smoke—all of which are relevant to wildfire monitoring. Sentinel-2 imagery is well suited for this purpose due to its high spatial detail, relatively frequent revisit rate, and access to spectral bands useful for fire analysis.

Fires were studied in three diverse geographic regions known for frequent and intense wildfire activity: California, southeastern Australia, and the Amazon rainforest. Each region presents distinct environmental characteristics that contribute to the robustness and generalizability of the training dataset. California offers a temperate Mediterranean climate with dry summers, mountainous terrain, and frequent wildland-urban interfaces—conditions that make it highly susceptible to rapidly spreading fires near human settlements. Australia, particularly the southeastern states like New South Wales and Victoria, experiences hot, dry summers and is covered by vast eucalyptus forests that are highly flammable and prone to extreme bushfires, including crown fires and pyroconvective events. The Amazon presents a tropical rainforest ecosystem with high humidity, dense vegetation, and seasonal dry periods. In this region, many fires are anthropogenic, often resulting from deforestation activities. By including these three regions, the dataset incorporates a wide range of fire behaviors, land cover types, atmospheric conditions, and fire causes, which helps improve the model's ability to generalize across different wildfire scenarios worldwide.

To retrieve Sentinel-2 imagery corresponding to the fire events, the GEE (Google Earth Engine) platform was used. GEE is a cloud-based geospatial analysis environment that provides access to a vast archive of satellite imagery and allows efficient querying based on spatial and temporal filters [70]. In this study, GEE was employed specifically to locate and download Sentinel-2 scenes that overlapped with the VIIRS fire detections. It was also used to filter out images with excessive cloud cover, ensuring that only visually usable scenes were selected. While the VIIRS fire data and subsequent image processing steps were handled locally, GEE significantly streamlined the process of identifying relevant Sentinel-2 imagery and exporting it for further analysis.

## 3.2 Patch Extraction

The construction of a high-quality dataset for wildfire detection began with parsing VIIRS fire records and identifying satellite imagery that spatially and temporally matched these events. Each VIIRS detection includes an estimate of confidence, which reflects the sensor's internal assessment of how likely the detected thermal anomaly corresponds to an actual fire. Only records with high confidence (i.e., labeled as "h") were retained to reduce noise and ensure the reliability of ground-truth labels. Additionally, a threshold of

FRP ≥ 50 MW was applied. FRP, or Fire Radiative Power, measures the radiative energy output of a fire in megawatts and serves as a proxy for fire intensity. Filtering based on FRP helps exclude weak or ambiguous thermal signals that may not result in visually detectable fire activity in the corresponding imagery.

Once relevant fire points were selected, the Sentinel-2 Surface Reflectance product (S2 SR HARMONIZED) was queried using GEE [71]. For each fire event, the system searched for available Sentinel-2 images within a ±24-hour time window and filtered out scenes with more than 70% cloud cover. This temporal window was necessary because Sentinel-2 does not provide continuous coverage–each satellite in the constellation revisits the same location approximately once every 5 days, though the effective revisit frequency can be improved to 2–3 days in mid-latitudes thanks to the dual-satellite configuration (Sentinel-2A and 2B). As a result, even if VIIRS detected a fire on a given day, a corresponding Sentinel-2 overpass may not have occurred at the same moment. Expanding the temporal range to ±24 hours increased the likelihood of capturing a usable image while maintaining a reasonable temporal correlation with the fire event.

If a suitable image was found, a 224×224 pixel patch (approximately 4480×4480 m at 20-meter resolution) centered on the fire coordinates was extracted. To avoid data leakage between training, validation, and test sets, only one patch per Sentinel-2 scene was retained. Although Sentinel-2 images cover large areas, extracting multiple nearby patches from the same scene could result in nearly identical samples appearing in different dataset splits, thereby artificially inflating performance metrics. Scene identifiers were tracked throughout the pipeline to prevent reusing the same image in different parts of the dataset.

The extracted patches, originally in GeoTIFF format, were downloaded from GEE and converted to .npz files–NumPy's compressed binary format for storing arrays. Each .npz file contains the image data and an associated binary label (fire or no fire). This format was chosen for its efficiency and compatibility with machine learning workflows in Python, particularly when handling large volumes of image data. During conversion, only three spectral bands were retained: Red (B4), Near-Infrared (B8), and Short-Wave Infrared 2 (B12). Details regarding the choice of spectral bands and normalization procedures are discussed in Section 3.3.

After preprocessing, a manual quality assurance (QA) step was performed on all fire-labeled patches. Patches were visually reviewed, and any samples with excessive cloud cover, sensor artifacts, ambiguous visual evidence, or no visible signs of fire were excluded. This step was crucial for ensuring that the model learned from clear, unambiguous examples of fire-related imagery and reduced the number of false labels caused by automated misalignment or misclassification.

To construct a balanced dataset, an equivalent number of non-fire patches was collected. These samples were drawn from the same geographic regions and seasonal periods, but were restricted to locations and times where no active VIIRS fire detections were present. This ensured that the model would learn to distinguish fire imagery not only from random backgrounds, but from visually similar, fire-prone areas during fire season. Importantly, non-fire patches were also subjected to manual review, with cloudy or corrupted images excluded to maintain dataset quality. The same preprocessing pipeline–band selection, normalization, resizing, and conversion to NumPy archive format (.npz) –was applied uniformly to both fire and non-fire samples.

In total, the dataset includes 6000 labeled patches: 3000 fire and 3000 non-fire. Table 2 summarizes the class balance together with the median FRP and residual cloud fraction of the accepted Sentinel-2 scenes.

Table 2. Regional composition of the dataset, typical FRP and residual cloudiness after filtering.

| **Region** | **Fire** | **Non-fire** | **Median FRP (MW) [min–max]** | **Median Cloud % [min–max]** |
|---|---|---|---|---|
| Amazon | 726 | 726 | 83.5 [50.0 - 522.7] | 1.15 [0 – 66.68] |
| Australia | 1799 | 1799 | 76.2 [50.0 - 362.4] | 0.35 [0 – 69.00] |
| California | 475 | 475 | 90.1 [50.6 - 343.2] | 1.85 [0.03 – 47.2] |
| Total | 3000 | 3000 | 83.9 [50.0 – 522.7] | 1.12 [0 – 69.00] |

The complete patch extraction and filtering process is summarized in Figure 3, which outlines the logic flow from initial fire detection through to final dataset generation. The resulting dataset, combining automated filtering, precise spatiotemporal alignment, and human validation, provides a solid foundation for training adaptive neural networks under realistic satellite imaging conditions.

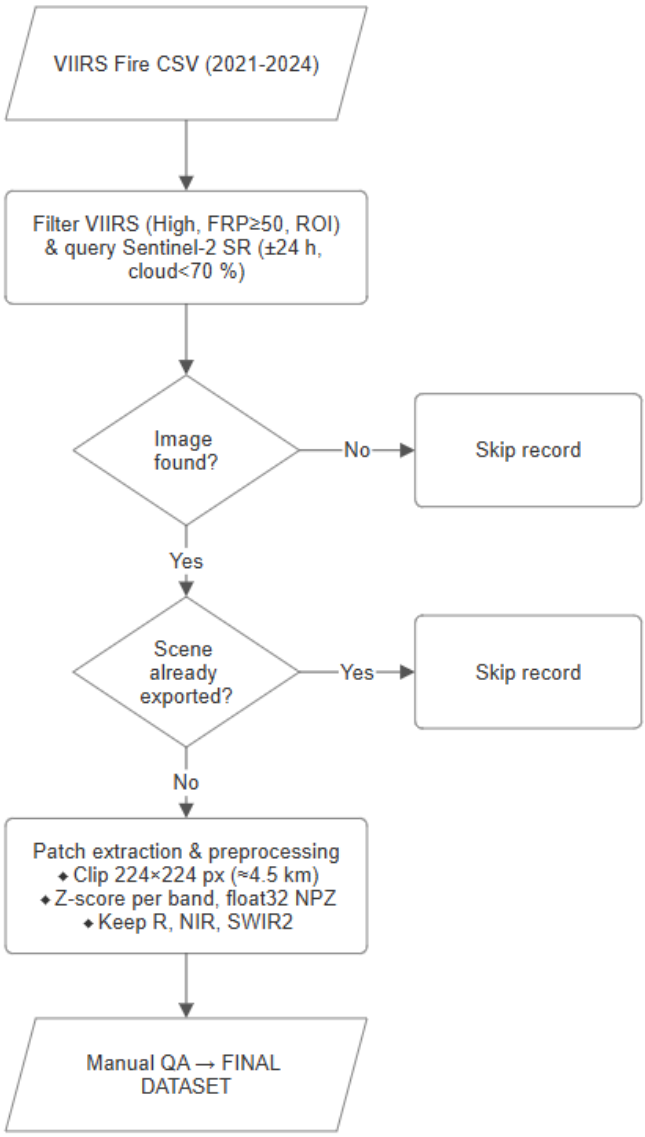



Figure 3. Patch extraction and preprocessing pipeline. Each VIIRS detection is matched to a single Sentinel-2 scene, clipped, normalized, converted, and manually verified before inclusion.

## 3.3 Channels and normalization

To balance energy efficiency and detection performance in an onboard setting, only a subset of three spectral bands from the Sentinel-2 imagery was used as input for model training. The selection focused on bands that provide complementary and informative signals for wildfire detection:

- B04 (Red, 665 nm) captures surface reflectance in the visible red spectrum and is effective for identifying burned areas and vegetation loss.
- B08 (NIR–Near Infrared, 865 nm) highlights vegetation health and contrast differences in smoke plumes and fire scars.

- B12 (SWIR2–Short-Wave Infrared, 2190 nm) responds strongly to active fire fronts and high-temperature surfaces, making it critical for identifying thermal anomalies [72].

This combination of Red, NIR, and SWIR2 bands was selected to balance discriminative power and computational efficiency. Including just these three channels allows the model to capture critical features of fire scenes–such as thermal activity, vegetation state, and smoke presence–while keeping the input dimensionality low. This design choice helps minimize memory usage and computation time, which is especially important for onboard inference on low-power CubeSat platforms.

All imagery was sourced from the Sentinel-2 Surface Reflectance (SR) product, which contains reflectance values corrected for atmospheric effects. As this product is already preprocessed and standardized, no additional radiometric correction was necessary. Prior to model training, each patch underwent per-band z-score normalization, a widely used method that improves training stability by standardizing input distributions [73]. For each of the selected bands, pixel values $x$ were transformed as follows:

$$x' = \frac{x - \mu}{\sigma}$$

where $\mu$ and $\sigma$ are the global mean and standard deviation for that band, computed from the full set of downloaded image patches. These statistical parameters were precomputed and embedded into the preprocessing script to ensure consistent normalization across the entire dataset.

An illustration of the input format is shown in Figure 4, which displays a wildfire example across three spectral combinations: standard RGB, the isolated SWIR2 band, and the Red+NIR+SWIR2 composite used as model input. As seen in the image, the RGB visualization obscures fire activity under smoke, whereas the SWIR2 band clearly reveals burning regions, especially when visualized using a thermal color map. The composite image (R+NIR+SWIR2) enhances both flames and burned vegetation, demonstrating the effectiveness of this configuration for wildfire detection.

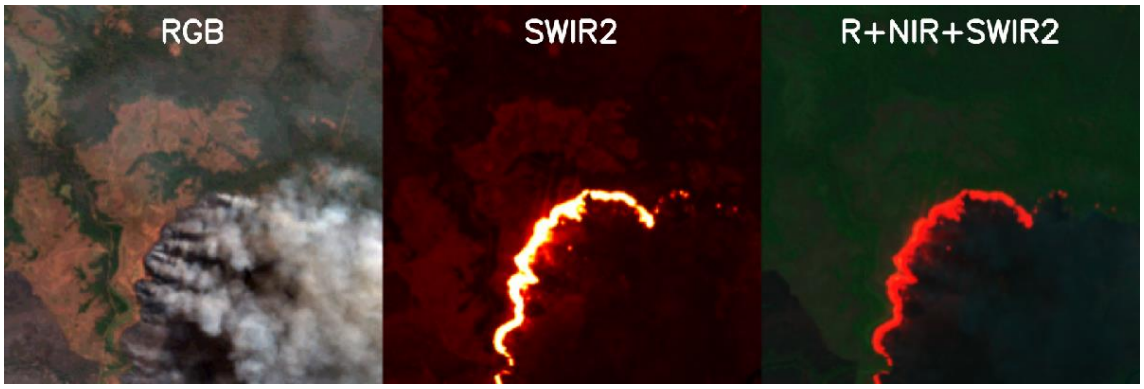



Figure 4. Example wildfire patch shown as RGB, SWIR2 (thermal), and the composite input used by the model (Red, NIR, SWIR2).

These preprocessing steps–band selection and normalization–were consistently applied to both fire and non-fire patches to ensure uniform input conditions during training. By limiting the input to three carefully chosen bands and applying statistically grounded normalization, the dataset achieves both computational efficiency and spectral relevance. This setup is particularly well-suited for training compact convolutional models intended for deployment on energy-constrained platforms such as CubeSats, where every byte of input data and every floating-point operation must be justified.

## 3.4 Augmentation

To improve the robustness and generalization of the wildfire detection model, data augmentation techniques were applied during training. Augmentation artificially increases dataset diversity by introducing controlled variations into the input images, allowing the model to better handle real-world variability such as orientation changes, lighting conditions, and minor noise. This is particularly valuable in satellite imagery tasks, where capturing a wide range of natural conditions (e.g., viewing angles, atmospheric effects) directly in the dataset can be challenging.

However, in the context of wildfire detection, augmentation must be applied with caution. Wildfires are typically small and sparse relative to the full image patch, and their spectral signature–particularly in the SWIR2 band–is subtle but critical. Overly aggressive transformations (e.g., large rotations, elastic warping, or synthetic noise) risk distorting or even removing the visual cues that signify a fire, which can degrade model performance or lead to inconsistent labeling. Therefore, the augmentation strategy in this study was deliberately kept moderate and spatially consistent.

During training, the following augmentations were applied using the Albumentations library [74]:

- Horizontal and vertical flipping (p=0.5 each) helps the model become invariant to fire orientation and landscape symmetry, especially useful since satellite images can come from varying orbital directions.
- Random 90-degree rotation (p=0.5) preserves image geometry while introducing orientation variability. Unlike arbitrary-angle rotations, 90° steps avoid interpolation artifacts and keep the sharpness of fire edges.
- Random brightness and contrast adjustment (±15%, p=0.5) simulates variability in illumination and atmospheric haze, making the model more tolerant to minor reflectance deviations without overwhelming the original fire signal.

No spatial cropping, noise injection, affine transforms, or color jittering were used, as such operations could compromise the visual integrity of the small-scale fire patterns. For validation and testing, no augmentations were applied beyond normalization. This ensures that evaluation reflects the model's performance on true, unaltered data and aligns with the conditions it would face during real-world deployment.

The applied augmentation strategy was carefully designed to enhance generalization without compromising detection sensitivity. By simulating realistic variations while preserving the fine-grained structure of fire regions, the model is better equipped to operate reliably across diverse geographic areas and environmental conditions.

In summary, this chapter presented a complete pipeline for constructing a wildfire detection dataset, covering data sourcing, spatial-temporal filtering, patch extraction, normalization, and carefully constrained augmentation. The resulting dataset is balanced across classes and regions, and optimized for use in onboard deep learning systems [75]. With the data foundation established, the next chapter introduces the neural network architectures and inference strategies designed to meet the energy and latency constraints of CubeSat platforms.

# 4 Adaptive CNN Architecture and Training Strategy

Designing an onboard wildfire detection model for CubeSats requires balancing high accuracy with strict energy and latency limits. This chapter introduces a neural network architecture that supports adaptive inference through early-exit branches. It first reviews the design requirements for deploying neural networks in low-power orbital platforms. It then describes the baseline architectures considered for the task, including compact and efficient CNN backbones suitable for onboard inference. A core innovation of the proposed system–the early-exit mechanism–is presented in detail, followed by the training procedure that ensures both high performance and efficient exit decision-making. The chapter concludes with deployment considerations for embedded GPU execution.

## 4.1 Design requirements

The goal of this work was to develop a convolutional neural network architecture capable of detecting wildfires from multispectral satellite imagery in real time on a resource-constrained CubeSat platform. To ensure practical deployability, the design process was guided by strict hardware and operational constraints, with a focus on computational efficiency, energy usage, and latency. All architectural and training decisions were shaped by the requirements of edge computing in low-power orbital environments.

As a reference deployment target, the NVIDIA Jetson Nano was selected, operating in its 5 W power mode. The Jetson Nano served as a realistic and accessible proxy for CubeSat-grade embedded hardware, offering a quad-core ARM Cortex-A57 CPU, a 128-core Maxwell GPU, and approximately 472 GFLOPs of theoretical compute throughput. The model runs directly in PyTorch and executes on the embedded GPU, making the platform naturally compatible with dynamic early-exit logic.

Other edge accelerators, such as the Intel Myriad X and Google Coral Edge TPU, were also considered. However, both platforms rely on static computation graphs (via OpenVINO 2022.3 and EdgeTPU Compiler v2), which complicates support for runtime branching. In the case of Myriad X, early-exit evaluation was still performed by manually partitioning the network into separate static sub-models and managing control flow externally from the host system. While technically feasible, this approach introduced additional complexity and coordination overhead, and was therefore not adopted as the

main deployment strategy. A detailed description of the Myriad X implementation and its limitations is provided in Section 5.5.

Real-time constraints were defined using a practical example scenario based on Sentinel-2 imagery. A full scene at 20 m resolution is approximately 5490 × 5490 pixels (≈109 × 109 km) in size. To enable fine-grained detection and minimize the risk of missing small or spatially localized fire signals between patch boundaries, a sliding-window approach was applied with a stride of 112 pixels. This resulted in 2401 overlapping patches of size 224 × 224 pixels, each to be processed individually by the CNN.

Assuming the satellite maintains ground contact with a receiving station for roughly 5 minutes (≈300 s), and allocating 180 seconds exclusively to onboard inference (with the remaining time reserved for data handling and transmission), the average processing time per patch must not exceed:

$$t_{per\ patch} = \frac{180\ s}{2401} \approx 75\ ms$$

Given the Jetson Nano's theoretical throughput of 472 GFLOPs/s, this implied a maximum per-patch budget of:

$$FLOPS_{worst} = 75\ ms * 472\ GFLOPSs/s \approx 35\ GFLOPs$$

To account for potential runtime variability, system interrupts, and memory contention, a safety margin of 2× was applied to the deepest (last) exit. Consequently, the maximum allowable compute footprint was constrained to <70 GFLOPs, while the average compute per patch was required to stay below 35 GFLOPs.

In addition to compute constraints, onboard memory is a critical limiting factor. While the Jetson Nano includes 4 GB of LPDDR4 RAM, much of this is consumed by the operating system, preprocessing pipelines, and especially the intermediate activation maps during inference. To avoid memory contention and ensure real-time operation, the total model parameter size must be tightly limited. Specifically, no more than 5% of available RAM–approximately 200 MB–was reserved for model weights, and an additional 2× safety margin was applied. This yielded a target upper bound of <100 MB for the model parameters. All architecture variants and branches must fit within this budget, including those with early-exit heads.

Finally, to ensure that architectural optimizations did not overly degrade detection performance, a minimum performance threshold was enforced: the final adaptive model was required to achieve at least 90% of the classification accuracy of a standard full-depth baseline CNN on the same dataset. This constraint ensured that efficiency gains were not achieved at the expense of real-world utility.

Together, these constraints defined the optimization space for designing and training the proposed adaptive CNN: maximize efficiency and reduce average inference time, without exceeding 70 GFLOPs per patch, without exceeding 100 MB of memory, and without sacrificing more than 10% of baseline accuracy (see Table 3 for a summary of system constraints). It is important to note that the design limits represent worst-case upper bounds derived from the communication window and onboard memory allocation. Satisfying these limits with a safety margin is desirable to accommodate unforeseen loads and to allow for the deployment of more complex models in the future.

Table 3. Summary of system-level constraints for adaptive CNN design on Jetson Nano

| **Constraint Type** | **Target Limit** | **Justification** |
|---|---|---|
| Max latency per patch | 75 ms | 5 min downlink window ÷ 2401 patches |
| Max per-patch FLOPs | 70 GFLOPs | Worst-case budget with 2× safety margin |
| Avg per-patch FLOPs | 35 GFLOPs | Needed to process entire scene in 180 s |
| Max model size | 100 MB | Fits within 5% RAM with 2× safety margin |
| Accuracy retention | 90% of full baseline | Ensures practical detection performance |
| Power draw | 5 W | Jetson Nano 5 W mode |

## 4.2 Baseline backbones

Selecting an appropriate convolutional backbone is a critical step in developing an efficient onboard wildfire detection model. The chosen architecture must balance classification performance with strict constraints on compute, memory, and power–while also offering structural flexibility to support adaptive inference with early exits.

To this end, four candidate architectures were evaluated as baselines:

- MobileNetV2

- MobileNetV3-Small
- MobileNetV3-Large
- ResNet-18

The MobileNet architectures are specifically designed for efficient inference on edge devices. They use depthwise separable convolutions (DW-Convs), inverted residual blocks, and lightweight activation functions (such as h-swish in V3), all of which significantly reduce the number of parameters and floating-point operations compared to conventional CNNs. These models are also fully open-source, well-supported by modern deep learning frameworks, and modular by design–making it straightforward to insert early-exit branches between convolutional stages. This modularity was a key criterion, as the early-exit mechanism requires internal classifiers to be placed at intermediate depths of the network without disrupting its structure or flow of information.

ResNet-18, while more computationally demanding, was included as a widely used standard CNN with strong representational power and well-established optimization behavior. Its inclusion served to benchmark MobileNet's efficiency-accuracy tradeoff against a classic baseline.

To objectively compare these models, all four architectures were pretrained on ImageNet and then fine-tuned on the wildfire dataset using an identical training pipeline (without early exits). The results are summarized in Table 4.

Table 4. Baseline model comparison on ImageNet and wildfire dataset [76], [77].

| **Model** | **Parameters** | **FLOPs (forward)** | **Top-1 Accuracy (ImageNet)** | **F1-Score (Wildfire)** | **Size (MB)** |
|---|---|---|---|---|---|
| MobileNetV3-Small | ~2.9 M | ~60 MFLOPs | 68.1% | 0.9571 | 5.9 |
| MobileNetV2 | ~3.4 M | ~300 MFLOPs | 71.8% | 0.9699 | 8.7 |
| MobileNetV3-Large | ~5.4 M | ~220 MFLOPs | 75.6% | 0.9640 | 16.2 |
| ResNet-18 | ~11.7 M | ~1.8 GFLOPs | 69.8% | 0.9588 | 42.7 |

Despite being relatively lightweight, MobileNetV2 consistently achieved the highest F1-score on the wildfire dataset. Its performance surpassed both more compact architectures (like V3-Small) and larger ones (like ResNet-18), while remaining well below the compute and memory constraints defined in Section 4.1. In addition, its architecture is

composed of 17 inverted residual blocks grouped into 7 stages, making it highly modular and well-suited for early-exit integration.

ResNet-18, despite having more than 5× the parameter size of MobileNetV2 and nearly 6× the model footprint, achieved a lower F1-score compared to MobileNetV2. This suggests that larger models do not necessarily translate into better performance on limited datasets–ResNet-18 may require more training data to fully leverage its capacity. However, even if more data were available, the accuracy gains would likely be marginal compared to the already strong performance of MobileNetV2, while incurring a significant penalty in weight size and memory consumption.

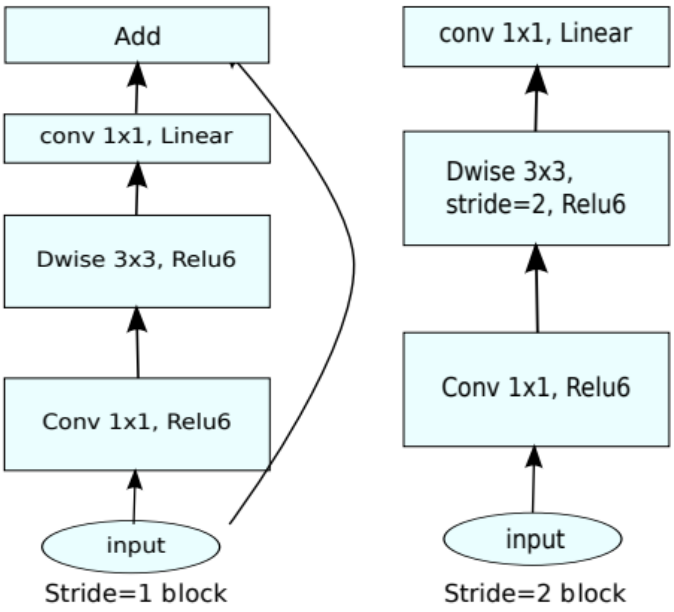
Given its strong empirical accuracy, compact size, and clean stage-wise modularity, MobileNetV2 was selected as the reference backbone for all subsequent analysis, early-exit placement, and architectural optimization.

### 4.2.1 Architecture overview

In a typical MobileNetV2 block, the input tensor is first expanded into a higher-dimensional space using a 1×1 convolution, followed by a 3×3 depthwise convolution for spatial filtering. The result is then projected back into a lower-dimensional embedding via another 1×1 convolution, this time without a non-linearity (i.e., using a linear activation). This pattern, where expansion precedes compression, is known as an inverted residual structure, in contrast to traditional residual blocks that compress first and expand later. When the input and output dimensions match and the stride is 1, a skip connection is added, enabling efficient gradient flow and improved feature reuse.

The linear bottleneck at the end of each block omits activation functions in the output projection, preserving feature expressiveness in low-dimensional spaces. This design reduces information loss caused by non-linearities in narrow channels–an important consideration in efficient networks.

Figure 5 illustrates the structure of MobileNetV2 blocks for both stride-1 and stride-2 variants. The left-hand diagram shows the residual connection in the stride-1 case, while the right-hand diagram depicts the downsampling variant without residual connection.




(d) Mobilenet V2

Figure 5. MobileNetV2 block structure. Left: stride-1 block with residual connection. Right: stride-2 block without skip connection. Each block expands, filters, and compresses the input using depthwise separable convolutions [36].

In the MobileNetV2-1.0 configuration, the full network consists of 17 inverted residual blocks, grouped into 7 stages based on spatial resolution. The network begins with a standard 3×3 convolution and ends with a final 1×1 convolution before global average pooling and classification. The input resolution of 224 × 224 pixels is progressively reduced to 7 × 7 in the final stage. The total parameter count is approximately 3.4 million, and a full forward pass requires about 300 million FLOPs [36].

Thanks to its clean modular layout and the separation between stages, early-exit classifiers were later inserted after selected downsampling stages, enabling conditional inference with minimal architectural modification.

## 4.3 Early-Exit Mechanism

A total of three exit heads were introduced, each positioned after a specific stage of the network that applies spatial downsampling. These stages divide the network into progressively deeper semantic levels, and their reduced activation resolutions make them efficient branching points for classification. In MobileNetV2, spatial downsampling

happens at the initial 3×3 conv layer and in the first block of Stages 2, 3, 4, and 6. However, the earliest two downsampling points (the initial conv layer and Stage 2), which together account for only ≈5% and ≈15% of total FLOPs and extract very low-level features, were excluded from early-exit placement. As a result, the early exits were placed after more semantically meaningful points:

- Exit 1–after Stage 3, following the 6th inverted residual block; resolution 28×28, 32 channels; ≈39% cumulative FLOPs.
- Exit 2–after Stage 5, following the 13th inverted residual block; resolution 14×14, 96 channels; ≈75% FLOPs.
- Exit 3–after Stage 7, following the final (17th) inverted residual block; resolution 7×7, 320 channels; full model depth. This third exit is the standard final output of MobileNetV2.

Figure 6 illustrates the modified architecture and highlights the positions of the early-exit heads.

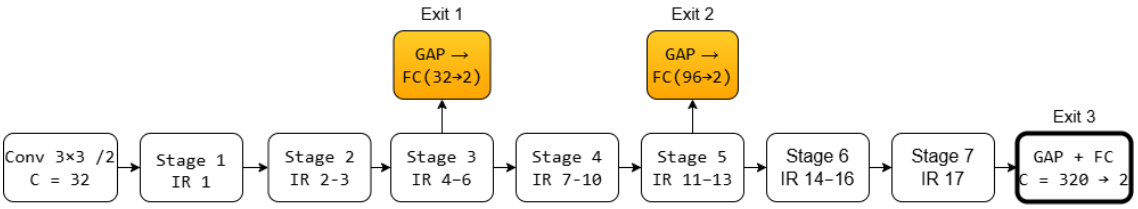



Figure 6. Structural overview of MobileNetV2-1.0 with integrated early-exit heads. Each head applies global average pooling followed by a fully connected layer for binary classification. IR = Inverted Residual block. GAP = Global Average Pooling.

Each exit head applies a simple classification structure:

- Exit 1: AdaptiveAvgPool2d(1) → Flatten→ Dropout(0.1) → Linear(C → 2)
- Exit 2: AdaptiveAvgPool2d(1) → Flatten→ Dropout(0.2) → Linear(C → 2)
- Exit 3: AdaptiveAvgPool2d(1) → Flatten → Dropout(0.5) → Linear(C → 2)

Where C is the number of input channels at each stage. At inference time, confidence is computed from softmax scores as:

$$p_{max} = max(softmax(\hat{y}_k))$$

Exits are checked sequentially. If $p_{\max} \geq \tau_k$ at exit $k$, the model halts and returns the prediction. Otherwise, it proceeds to the next exit.

The confidence thresholds $\tau_1$ and $\tau_2$ are not hard-coded into the model; instead, they are stored as run-time attributes. This allows them to be adjusted dynamically during operation, without requiring the model to be recompiled or re-exported. As a result, the exit behavior can be tuned in real time–such as lowering thresholds under severe energy constraints or raising them in high-alert conditions.

The final exit always produces a prediction, ensuring completeness. To train all exits simultaneously, a joint loss function is used, aggregating cross-entropy losses from each exit head with weighted contributions [78]:

$$\mathcal{L} = \sum_{k=1}^{3} \alpha_k CE(y, \hat{y}_k), \qquad \alpha = (0.3, 0{,}3\ ,04)$$

Here, $CE(y, \hat{y}_k)$ is the cross-entropy between the ground truth label $y$ and the prediction $\hat{y}_k$ from exit $k$, and $\alpha$ determines the relative weighting. Slight emphasis is given to the final classifier to encourage high-capacity learning while still supervising the early classifiers. The weights were chosen heuristically to ensure that early exits receive sufficient gradient signal for meaningful learning, while prioritizing accuracy at the final output. The values sum to 1, ensuring a balanced contribution to the overall loss.

Table 5 summarizes the structural placement and resource footprint of each early-exit head. The Depth column indicates the relative position of the exit within the 17-block MobileNetV2 backbone. Cumulative FLOPs reflect the total compute required to reach each exit during inference. The Total Params column shows the number of model parameters if inference were to terminate at that point, while Δ Params represents the additional parameter cost introduced by the exit head itself. These metrics provide insight into how computational and memory demands grow with each successive exit stage.

Table 5. Exit head locations and associated resource cost

| **Exit k** | **Depth** | **Cum. FLOPs (GF)** | **Total Params (kB)** | **Δ Params (kB)** |
|---|---|---|---|---|
| Exit 1 | 6/17 (≈35%) | 0.11 | 208.1 | 0.3 |
| Exit 2 | 13/17 (≈76%) | 0.21 | 2106.4 | 0.8 |
| Exit 3 | 17/17 (100%) | 0.30 | 7123.2 | 2.6 |

This lightweight early-exit design allows the network to adaptively allocate computation based on input difficulty, improving average efficiency without compromising overall accuracy.

## 4.4 Training Procedure

The adaptive wildfire detection model was trained using supervised learning on the multispectral patch dataset described in Chapter 3. The goal was to optimize all classifier heads jointly while maintaining computational and accuracy constraints defined in earlier sections.

The dataset was split into training, validation, and test subsets in a 70/15/15 ratio with stratified sampling to preserve class balance. Each sample consisted of a 224×224 image patch with three input channels–Red, Near-Infrared (NIR), and Short-Wave Infrared (SWIR2)–preprocessed and normalized as described in Section 3.3. Augmentations were applied only during training (see Section 3.4) and excluded from validation and test sets.

The model architecture was based on MobileNetV2-1.0 with weights pretrained on ImageNet. This initialization accelerated convergence and improved performance, particularly for the deeper layers. The three early-exit classifier heads were appended to intermediate feature maps and were initialized with random weights.

Training was conducted on a single NVIDIA RTX A4000 GPU using PyTorch. The code used for preprocessing, training, and evaluation of the model is publicly available in a dedicated GitHub repository [79]. The optimizer used was AdamW with a learning rate of $1.5\times10^{-4}$, a batch size of 16, and weight decay of 1e–4. A cross-entropy loss with label smoothing (0.1) was minimized across all classifier heads. The weighted joint loss function was applied, using weights $\alpha = (0.3,0.3,0.4)$ to balance learning across the three exits. A maximum of 100 epochs was allowed, but early stopping with a patience of 10 epochs was used to avoid overfitting, based on validation F1-score from the deepest (final) classifier. In practice, training typically converged between epochs 30 and 40, depending on the initialization and random split.

Table 6. Core hyperparameters used during model training

| **Parameter** | **Value** |
|---|---|
| Optimizer | AdamW |
| Learning rate | 1.5e-4 |
| Batch size | 16 |
| Epochs (max) | 100 |
| Early stopping patience | 12 |
| Weight decay | 1e-4 |
| Loss function | CrossEntropy + smoothing (0.1) |
| Joint loss weights $\alpha$ | (0.3, 0.3, 0.4) |
| Backbone initialization | ImageNet pretrained |
| Exit heads initialization | Random |
| Train/Val/Test split | 70% / 15% / 15% |
| Input size | 224×224 px |
| Input channels | R + NIR + SWIR2 |
| Compute device | NVIDIA RTX A4000 |

After training, the final model is executed directly using the PyTorch runtime, without exporting to external inference frameworks. This choice preserves support for dynamic control flow, which is essential for implementing confidence-based early exits. Other formats such as TorchScript or TensorRT require static computation graphs and therefore do not support runtime branching. For example, deploying the model in TensorRT would require exporting each exit head as a separate engine and manually coordinating execution–an approach that adds significant complexity and does not align with the design goals of flexible inference.

Although PyTorch supports post-training INT8 quantization, this feature is primarily optimized for x86 CPUs via back-ends such as QNNPACK and FBGEMM. On Jetson Nano, INT8 operations are not natively accelerated and are often emulated, leading to negligible or even negative performance impact. As a result, inference is performed using FP16 precision on the GPU, which provides a practical balance between speed, accuracy, and power efficiency. For these reasons, the final model is retained in its original PyTorch format and executed in mixed precision (FP16) during deployment.

This chapter presented the complete design and training strategy of the proposed adaptive convolutional neural network for onboard wildfire detection. A lightweight MobileNetV2 backbone was selected for its modularity and efficiency, with early-exit heads integrated at strategic points to enable dynamic inference based on input difficulty. Design choices were shaped by realistic hardware constraints, including strict compute and memory budgets modeled after CubeSat-grade devices. The network was trained jointly using a weighted multi-exit loss, with all classifiers optimized simultaneously under a unified supervision scheme. The final model was executed in FP16 precision directly within the PyTorch runtime, ensuring compatibility with the embedded GPU and preserving support for dynamic early-exit logic.

The next chapter evaluates this architecture in practice, reporting accuracy, latency, and energy consumption under various configurations and thresholds. Special attention is given to the trade-offs between speed and performance, as well as to the behavior of early-exit branches across diverse test scenarios.

# 5 Experimental Results

This chapter presents a comprehensive evaluation of the proposed adaptive wildfire detection system. The analysis covers both classification performance and operational efficiency, focusing on how early-exit strategies affect accuracy, latency, and energy consumption. The results are organized into several categories, including baseline comparisons, threshold ablations, exit-wise behavior, and real-world performance on embedded hardware.

The evaluation follows a structured protocol that considers both model-level and hardware-level metrics. Specifically, it assesses how the adaptive model performs relative to its full-depth baseline in terms of accuracy and computational cost, how different confidence thresholds influence exit distribution and detection outcomes, and how inference behaves across various deployment platforms such as the Jetson Nano and Intel Myriad X.

## 5.1 Test set and metrics

All experiments were conducted on a held-out test set comprising 15% of the full dataset, stratified by class and region. The resulting test set contains 900 image patches, with equal representation of fire and non-fire samples.

To assess classification performance, the following standard metrics were computed:

- Accuracy–the overall proportion of correctly classified patches.
- Precision–the fraction of predicted fire patches that are actually fire.
- Recall (True Positive Rate) –the fraction of actual fire patches that are correctly detected.
- F1-score–the harmonic mean of precision and recall.

Among these, recall is prioritized as the most critical metric for the intended application. In the context of wildfire detection, false negatives (i.e., undetected fires) are far more costly than false positives. A missed detection may result in delayed response or unmitigated fire spread, with severe environmental and economic consequences. Therefore, all models and configurations are evaluated with an emphasis on minimizing fire-related false negatives, even at the expense of a slight increase in false alarms.

## 5.2 Threshold sweep on Jetson Nano

The adaptive inference mechanism relies on two confidence thresholds, $\tau_1$ and $\tau_2$, which control the decision to exit early at Exit 1 or Exit 2, respectively. These thresholds directly influence the trade-off between inference accuracy and computational efficiency. Lower values promote early exits, reducing latency and energy at the cost of possible misclassifications. Conversely, higher thresholds defer more predictions to deeper stages, increasing precision and recall but also computational cost.

To explore this trade-off, a grid search was conducted across:

$$\tau_1, \tau_2 \in \{0.50, 0.55, \ldots, 0.95\}$$

for a total of 100 unique threshold pairs. For each configuration, the following metrics were recorded on the Jetson Nano (5 W mode):

- Classification metrics: Accuracy, Precision, Recall, F1-score, ROC AUC
- Latency metrics: Mean latency per patch, 95th percentile latency
- Efficiency metrics: Average power (W), energy per patch (J)

The overall behavior of the system under varying thresholds is illustrated in Figure 7, which highlights the trade-off between speed, F1-score, and energy consumption. Three top-performing configurations are annotated directly on the plot.

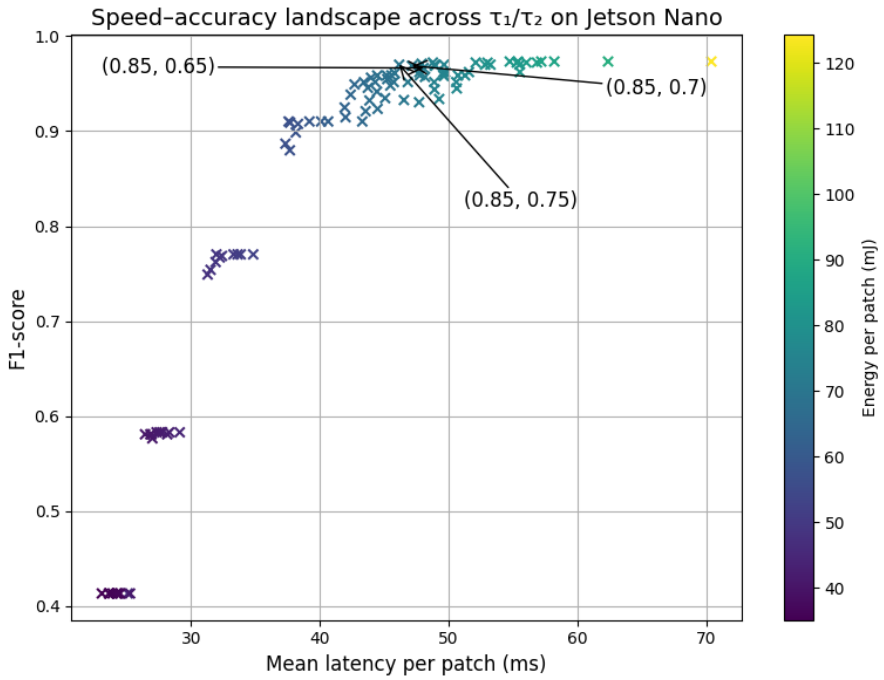



Figure 7. Speed–accuracy landscape across $\tau_1/\tau_2$ on Jetson Nano. Each point represents a threshold configuration. X-axis: mean latency (ms). Y-axis: F1-score. Color encodes energy consumption. Top-3 configurations are labeled.

At extreme threshold values $\tau_1 = \tau_2 = 1.0$, early exits are never triggered, all inputs pass through the full network. In this setting, the average per-patch latency is 70.4 ms, and the average energy consumption is 124.3 mJ.

At the opposite end of the spectrum, when $\tau_1 = \tau_2 = 0.5$, all predictions exit at the first classifier. This reduces the latency to 25.1 ms and the energy to 45.6 mJ, corresponding to a 64.3% latency reduction and a 63.3% energy reduction compared to the full-depth baseline.

Between these endpoints, both latency and energy decrease smoothly with lower thresholds, with the steepest savings occurring when $\tau_1 \leq 0.80$. This indicates that many clearly negative (non-fire) samples can be confidently handled in the early layers, avoiding the need for deeper processing.

While early exits improve speed and efficiency, they inevitably affect classification accuracy. With no early exits ($\tau = 1.0$), the model reaches 97.33% accuracy. At the most aggressive threshold ($\tau = 0.5$), accuracy drops sharply to 60.9%, as all decisions are made using shallow features. However, with moderate thresholds ($\tau = 0.75$), accuracy

remains high at ≈95.9%, suggesting that early exits can significantly reduce inference cost while preserving most of the model's classification ability.

## 5.3 Selection of operating point

To identify optimal threshold configurations, all threshold pairs were filtered to retain only those achieving at least 96.33% accuracy (i.e., within 1 percentage point of the full-depth baseline). Among these, the top 10 combinations were selected and ranked by energy savings, with latency as a secondary criterion. Table 7 summarizes their classification metrics and resource reductions relative to the baseline.

Table 7. Top 10 threshold configurations ranked by energy savings

| $\tau_1$ | $\tau_2$ | Accuracy | Latency (s) | Energy (J) | Δ Latency | Δ Energy | Δ Acc. |
|---|---|---|---|---|---|---|---|
| 0.85 | 0.75 | 97.00% | 0.0461 | 0.0701 | –34.42% | –43.62% | –0.33 |
| 0.85 | 0.70 | 96.89% | 0.0472 | 0.0716 | –33.01% | –42.42% | –0.44 |
| 0.85 | 0.65 | 96.67% | 0.0481 | 0.0721 | –31.73% | –41.98% | –0.66 |
| 0.90 | 0.70 | 97.11% | 0.0479 | 0.0732 | –31.94% | –41.14% | –0.22 |
| 0.90 | 0.65 | 96.89% | 0.0489 | 0.0739 | –30.46% | –40.50% | –0.44 |
| 0.90 | 0.75 | 97.22% | 0.0488 | 0.0741 | –30.70% | –40.42% | –0.11 |
| 0.95 | 0.65 | 97.00% | 0.0489 | 0.0750 | –30.50% | –39.67% | –0.33 |
| 0.85 | 0.80 | 97.00% | 0.0496 | 0.0750 | –29.47% | –39.63% | –0.33 |
| 0.90 | 0.80 | 91.22% | 0.0521 | 0.0792 | –25.97% | –36.24% | –0.11 |
| 0.85 | 0.85 | 97.00% | 0.0528 | 0.0803 | –24.94% | –35.42% | –0.33 |

Among these candidates, the best overall balance between accuracy and efficiency was achieved with the threshold pair $\tau_1 = 0.85, \tau_2 = 0.75$. This configuration yielded an overall accuracy of 97.0%, only –0.33 percentage points below the full model. Importantly, recall remained exceptionally high at 96.8%, corresponding to 435 correctly identified fire patches out of 450 in the test set. This represents a drop of only 0.3 percentage points compared to the full-depth recall of 97.1%. Precision was also strong at 97.1%, indicating very few false alarms.

Further reduction of thresholds produced diminishing returns. For instance, the configuration $\tau_1 = \tau_2 = 0.70$ achieved 38.31% latency savings and 46.77% energy savings relative to the full model. However, this came at the cost of a –2.11 pp drop in accuracy, and a more concerning reduction in recall to 93.7%, which corresponds to 42

missed fires out of 450. For high-stakes applications like wildfire detection, such a decline in sensitivity may be unacceptable.

Nevertheless, lower-threshold configurations like (0.70, 0.70) may still be considered in extreme power-constrained scenarios. One potential application is adaptive thresholding, where exit criteria are dynamically adjusted over time in response to onboard resource availability. For example, as battery capacity degrades or solar exposure diminishes, the system could gradually lower its thresholds to conserve energy–sacrificing some precision and recall in exchange for sustained operation.

Notably, the best-performing configurations tend to favor higher values of $\tau_1$ compared to $\tau_2$. This design choice reflects the role of Exit 1 as an early checkpoint operating on relatively shallow features. At this stage, feature representations may still lack sufficient semantic context for confident decisions–particularly in challenging cases with smoke, haze, or visually subtle fires. By using a higher $\tau_1$, the model avoids premature exits on uncertain inputs and instead defers such decisions to deeper layers, where features are more expressive. This asymmetric thresholding strategy allows the system to maintain high recall and precision, while still benefiting from early exits on clear-cut cases like clean non-fire scenes.

## 5.4 Exit-level analysis

Exit-wise performance was evaluated on the 900-patch test set using the selected thresholds $\tau_1 = 0.85, \tau_2 = 0.75$. For each exit point, Table 8 reports the number of samples that exited, along with F1-score, precision, recall, and accuracy achieved by that classifier. This allows for a detailed view of how prediction quality and workload are distributed across the three exit heads.

Table 8. Exit-wise distribution and performance

| **Exit k** | **Depth GFLOPs** | **# Samples** | **F1-score** | **Recall** | **Precision** | **Accuracy** |
|---|---|---|---|---|---|---|
| 1 | 0.11 | 181 | 87.80% | 90.00% | 85.71% | 97.71% |
| 2 | 0.21 | 529 | 98.27% | 99.05% | 97.51% | 97.99% |
| 3 | 0.30 | 190 | 95.02% | 92.11% | 98.13% | 91.79% |
| Total | — | 900 | 97.00% | 96.89% | 97.10% | 97.00% |

A majority of the samples (59%) exited at Exit 2, indicating that this mid-depth classifier is responsible for the bulk of predictions. Exit 1 accounted for 20% of decisions, showing that a substantial number of clear-cut cases–likely non-fire images–can be confidently handled with very shallow features. The remaining 21% required full-depth processing via Exit 3, which typically corresponds to visually ambiguous or difficult cases. This distribution is visualized in Figure 8, which shows the proportion of samples handled by each exit head.

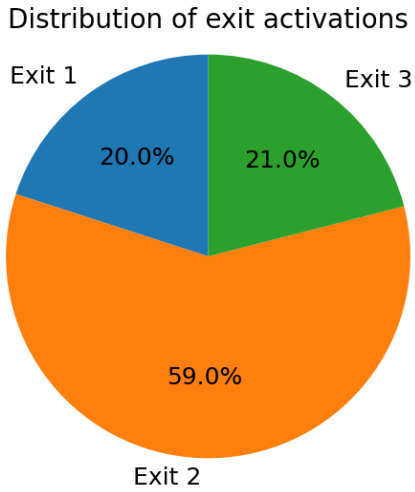



Figure 8. Distribution of exit activations

Exit 2 delivers the strongest recall at 99.05%, meaning it captures nearly all fire cases that reach it. Exit 1 performs slightly worse in terms of recall (90%), which is expected given its limited semantic depth. However, by deferring uncertain samples to deeper stages, the system compensates for these early errors. Overall recall across all exits remains high at 96.89%, with only 15 fire patches missed out of 450.

Interestingly, Exit 1 and Exit 2 both achieve very high accuracy–97.7% and 98.0%, respectively–despite being significantly shallower than the final classifier. In contrast, Exit 3, which receives the hardest samples, shows lower accuracy (91.8%), as expected. This suggests that the early-exit mechanism successfully filters easy samples forward while escalating only the challenging ones to deeper layers.

The overall distribution results in a substantial computational saving. With only 21% of samples reaching the full network, the average inference workload is far below the worst-

case path, enabling more efficient real-time deployment. The mean per-patch compute is estimated as:

$$AvgFLOPs = \frac{N_1 * F_1 + N_2 * F_2 + N_3 * F_3}{N_{total}}$$

Substituting values:

$$AvgFLOPs = \frac{181 * 0.11 + 529 * 0.21 + 190 * 0.30}{900} \approx 0.2089\ GFLOPs$$

Compared to the full-depth model's 0.30 GFLOPs per patch, this represents a 30.4% reduction in average inference cost.

### 5.4.1 Qualitative Examples

To further illustrate the behavior of the early-exit mechanism, Figure 9 presents qualitative examples of input patches that exited at different stages of the network under the selected thresholds.


Figure 9. Representative patches routed to different exits

The leftmost patch corresponds to a fire case that exited at the earliest classifier. A clear and intense fire front is visible, with strong activation in the SWIR2 band, allowing the model to confidently classify it as a fire using only shallow features. In contrast, the center image shows a patch that triggered the final (Exit 3) classifier. Here, the fire signal is faint and localized, with limited contrast against the background, which likely made earlier classifiers uncertain and required deeper semantic processing for a reliable decision.

The rightmost patch illustrates a non-fire example that exited at the second classifier. Although no active flames are present, the image contains characteristics of burned vegetation–often associated with post-fire conditions. This visual ambiguity likely caused

the model to defer the decision past Exit 1, but confidently resolve it as non-fire at Exit 2 after incorporating broader context. Together, these examples demonstrate how the system adjusts its computational depth in response to input difficulty, routing unambiguous cases early while reserving deeper reasoning for ambiguous or low-signal inputs.

## 5.5 Myriad X feasibility test

Although Jetson Nano is the primary deployment platform in this work, an additional feasibility test was performed on the Intel Myriad X accelerator. While Myriad X does not support dynamic branching in OpenVINO, it remains attractive for space-based applications due to its low power profile and proven flight history.

To simulate early-exit behavior, the MobileNetV2 model was manually split into three sub-models at the same internal points used for early-exit placement. Each sub-model was exported to ONNX and then converted to OpenVINO IR format. At runtime, the host system orchestrated inference by sequentially running sub-models: the input patch was first processed by the initial segment, and if the confidence threshold was met, prediction terminated early. Otherwise, intermediate outputs were passed to the next sub-model, continuing until classification was complete.

A full threshold sweep–identical to that on Jetson Nano–was conducted. The full-depth baseline (single graph) achieved 95.8% accuracy, 0.038 s latency, and 0.029 J energy. Table 9 lists the top early-exit configurations with ≤ 1 pp accuracy drop. However, unlike on Nano, latency and energy increased, due to USB overhead from repeated host–device transfers.

Table 9. Top Myriad X configurations (compared to 95.8% full-model baseline)

| $\tau_1$ | $\tau_2$ | Accuracy | Latency (s) | Energy (J) | Δ Latency | Δ Energy | Δ Acc. |
|---|---|---|---|---|---|---|---|
| 0.70 | 0.70 | 94.78% | 0.0425 | 0.0339 | +11.54% | +15.86% | –1.00 |
| 0.75 | 0.55 | 96.22% | 0.0426 | 0.0339 | +11.92% | +15.96% | +0.44 |
| 0.75 | 0.60 | 96.56% | 0.0428 | 0.0341 | +12.26% | +16.51% | +0.78 |

Interestingly, several threshold combinations slightly outperformed the full model in terms of accuracy, despite their multi-stage execution. This is likely a side effect of the model partitioning process: exporting each segment individually to ONNX and

OpenVINO can affect the internal structure of the graph, memory layout, or numerical behavior. These differences may result in subtly altered inference paths or operator-level optimizations, which in turn could affect final prediction outcomes.

These findings suggest that early-exit architectures are not well-suited for deployment on Myriad X in their current form. The need to partition the model and coordinate execution externally introduces significant overhead that negates the potential efficiency gains. As a result, deploying a full-depth, single-graph model remains the most practical and performant option on this platform. Early-exit mechanisms are only advantageous on hardware that supports conditional branching within a single computation graph and offers low-latency model execution without frequent host–device synchronization.

# 6 Discussion and Evaluation

The experiments conducted on the Jetson Nano platform demonstrate that the proposed early-exit architecture achieves a strong balance between classification accuracy and computational efficiency. Using the selected confidence thresholds $\tau_1 = 0.85, \tau_2 = 0.75$, the adaptive model reached an F1-score of 97.0%, nearly matching the full-depth baseline, while reducing average inference latency by 34% and energy consumption by 44% per patch. The detailed metric comparison is presented in Table 10.

Importantly, the recall remained high at 96.9%, ensuring that the model successfully detects nearly all fire cases. This is particularly relevant for emergency response applications, where false negatives may result in serious consequences. The ability to maintain such a high recall, even under reduced compute budgets, confirms the suitability of the system for real-time, energy-constrained onboard environments such as CubeSats.

Table 10. Performance comparison between full-depth and early-exit models on Jetson Nano

| **Model** | **Accuracy** | **Recall** | **F1-score** | **Precision** | **Mean latency (s)** | **Energy per patch J** |
|---|---|---|---|---|---|---|
| Full model | 97.33% | 97.11% | 97.33% | 97.10% | 0.0704 | 0.1243 |
| Early exit model | 97.00% | 96.90% | 97.00% | 97.54% | 0.0461 | 0.0701 |

Despite their reduced depth, early classifiers achieved strong predictive performance: Exit 2, which handled the majority of cases, reached an F1-score of 98.0% and recall of 99.1%, outperforming even the final head in some metrics. This suggests that early exits effectively capture easy-to-classify samples while routing harder cases deeper into the network.

The distribution of per-patch inference time is visualized in Figure 10, which shows a stacked histogram of latency across exit heads. The long tail in latency is formed exclusively by samples processed to Exit 3, while the majority of patches are resolved in significantly shorter time. This illustrates how conditional computation adapts to input difficulty and contributes to overall efficiency.

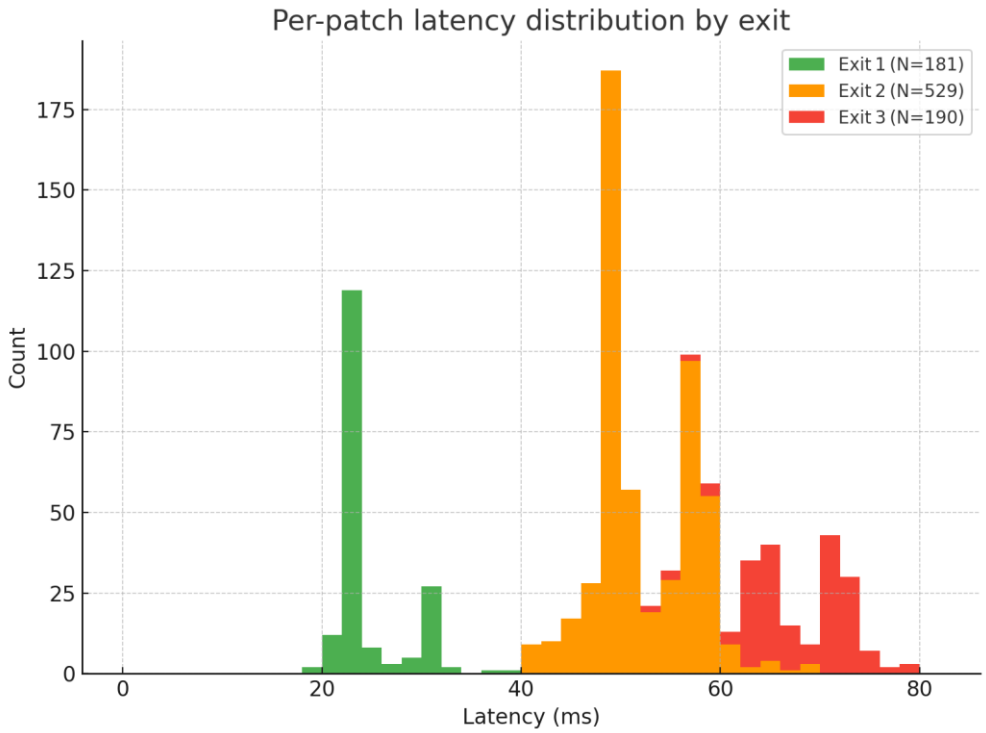



Figure 10. Latency distribution per exit ($\tau_1 = 0.85$, $\tau_2 = 0.75$). Exit 1 and 2 handle 80% of patches within 30 ms, longer latencies are due solely to Exit 3 processing.

## 6.1 Compliance with Design Requirements

The proposed system was designed to meet a set of strict deployment constraints defined in Section 4.1, including latency, compute, memory, power, and accuracy limits appropriate for onboard operation in CubeSat-class missions. Table 11 compares the actual performance of the selected early-exit configuration ($\tau_1 = 0.85, \tau_2 = 0.75$) against these original requirements.

All constraints were met with comfortable margins:

- The mean inference latency was 46.1 ms, well below the 75 ms threshold (a 39% margin).
- The average compute per patch was 0.2089 GFLOPs, significantly under the 35 GFLOPs budget. This large margin is primarily due to the lightweight nature of the MobileNetV2 backbone itself. While the early-exit mechanism contributes additional savings by allowing some predictions to terminate early, the majority of the efficiency stems from the base architecture being far below the theoretical maximum compute budget of the Jetson Nano.

- The measured power consumption of the full Jetson Nano module during inference was 3.2 W, remaining within the 5 W system-level power cap.
- The total model size was 7.3 MB, far below the 100 MB maximum allocated for weights.
- The overall accuracy of 97.0% comfortably exceeds the 90% target.

These results confirm that the system satisfies all operational constraints for real-time, onboard inference, with ample headroom for further scaling, on-device postprocessing, or long-duration deployments.

Table 11. Compliance with design constraints (Jetson Nano, 5 W mode)

| **Metric** | **Target** | **Achieved** | **Margin** |
|---|---|---|---|
| Accuracy | ≥ 90% | 97.0% | +7.0 pp |
| Latency (ms) | ≤ 75 ms | 46.1 ms | –28.9 ms (–39%) |
| Avg. FLOPs | ≤ 35 GFLOPs | 0.2089 GFLOPs | –34.79 GFLOPs (–99.4%) |
| Power | ≤ 5 W | 3.2 W | –1.8 W (–36%) |
| Model size | ≤ 100 MB | 7.3 MB | –92.7 MB (–92.7%) |

## 6.2 Operational Implications for a 6U CubeSat

To assess the feasibility of onboard deployment, the system's resource usage must be evaluated in the context of a typical CubeSat mission profile. As established in Section 4.1, a full Sentinel-2 scene is divided into 2,401 overlapping patches. Using the per-patch performance measurements from Section 6.3, the total compute time and energy required to process a full scene amount to 110.7 seconds and 168.1 joules, respectively. The corresponding average power draw during inference is 1.52 W, referring specifically to the compute workload of the model. This fits comfortably within the operational budget of most CubeSat platforms.

These values indicate that scene-level inference remains well within the constraints of a CubeSat-class system. As reviewed in the literature (Section 2.3), commercial 6U buses such as Open Cosmos typically budget 7–20 W for payload power and permit ~15% of each 90-minute orbit for full CPU or GPU usage–equivalent to ≈810 s of compute time per orbit.

Given this allowance, the system can process:

$$N_{scenes} = \frac{810\ s}{110.7\ s} \approx 7.3$$

Thus, the model is capable of analyzing up to seven full Sentinel-2 scenes per orbit without violating power or duty-cycle constraints. These figures confirm that the system is not only computationally efficient but also operationally viable for deployment on modern CubeSat platforms.

## 6.3 Error analysis and qualitative review

While the overall classification performance of the model is high, a closer inspection of failure cases provides insight into its current limitations and potential directions for improvement. Figure 11 presents three representative misclassifications from the test set, highlighting different sources of error.


Figure 11. Examples of false negatives (FN) and a false positive (FP). Left: FN at early exit; Center: FN at final exit; Right: FP (non-fire patch misclassified as fire).

The leftmost image shows a fire patch that was incorrectly classified as non-fire at Exit 2. The fire signal is very faint, partially obscured by a thin cloud layer, and only weakly visible even in the SWIR2 band. This suggests that low-contrast conditions can still cause under-detection, especially in early layers.

The center image is another false negative, but this one was passed through the full model and classified at Exit 3. The scene contains numerous scattered hot spots, likely representing low-temperature smoldering areas, but with little clear burned soil or thermal intensity. The model failed to aggregate these diffuse cues into a confident fire prediction.

On the right, a false positive is shown: a non-fire patch misclassified as fire. This scene includes an urban area, and a bright object in the center–possibly a reflective roof or

saturated pixel–appears to have triggered a spurious fire detection. While rare, such outliers highlight the importance of robustness to high-albedo surfaces and anthropogenic artifacts.

Together, these cases point to two dominant error sources: weak or ambiguous fire signatures, and confusion with visually similar non-fire features. These insights set the stage for broader architectural and dataset-level improvements discussed next.

## 6.4 Discussion

The proposed adaptive CNN achieves a consistent and meaningful efficiency gain–reducing average compute load by 30–40% while preserving a high recall rate of nearly 97% on wildfire detection tasks. This indicates that the majority of non-event images can be confidently classified early in the network, validating the hypothesis that dynamic inference is particularly well suited for Earth observation scenarios with sparse target events. Importantly, this gain does not come at the expense of mission-critical sensitivity, as fire-related false negatives remain minimal across all tested configurations.

Another strength of the approach lies in its flexibility through dynamic threshold control. Operators can adjust early-exit confidence levels in response to available power, operational mode, or mission priorities. For instance, during eclipse periods or battery degradation, thresholds can be lowered to favor energy savings, while in high-alert modes (e.g., during known fire season overpasses), thresholds can be raised to prioritize accuracy. This adaptability ensures that the system can gracefully balance performance and energy constraints over the satellite’s operational lifetime.

However, the model's performance is bounded by several limitations. First, the training dataset–while geographically diverse–remains limited in size, potentially reducing generalizability to unseen regions or seasonal conditions. Second, the network was restricted to only three spectral channels (Red, NIR, SWIR2), chosen for efficiency, but this excludes other potentially useful bands (e.g., mid-wave infrared or thermal channels). Lastly, no quantization was applied during deployment due to hardware limitations on the Jetson Nano. As a result, inference was performed in FP16 rather than more energy-efficient INT8 or 8-bit formats.

Future work could address these limitations by expanding the dataset (including temporal and cross-seasonal data), integrating additional spectral bands via attention-based input selection, and exploring model quantization and pruning for even greater energy efficiency. Moreover, deployment on newer hardware (e.g. radiation-hardened FPGAs with support for conditional execution) could unlock more aggressive optimizations for space-based AI inference.

# 7 Summary

This research set out to show that a miniature satellite can do more than passively relay images–it can actively decide, in real time and onboard, whether a wildfire is present in the scene below. To enable this, a dataset of 6,000 Sentinel-2 image patches was compiled–balanced evenly between confirmed fire and non-fire samples, and drawn from California, Australia, and the Amazon. All patches were filtered for cloud and sensor artifacts and reduced to just three spectral bands (Red, NIR, and SWIR-2) to preserve thermal information while fitting within CubeSat memory limits. This dataset served as the basis for developing an adaptive convolutional model capable of making real-time decisions under resource constraints.

At the core of the system is a MobileNetV2 backbone equipped with two early-exit heads. During inference, the network assesses its confidence after a shallow and an intermediate stage, and only proceeds to the final classifier when needed. Thresholds of $\tau_1 = 0.85$ and $\tau_2 = 0.75$, selected from a 100-point grid search, provided the best balance between responsiveness and accuracy–allowing fast classification of clear-cut cases while escalating ambiguous ones. This strategy worked as intended: 20% of test patches exited at the first classifier, 59% at the second, and only 21% required full-depth inference. Such conditional processing is the foundation of the model's efficiency.

Quantitatively, the early-exit configuration matched the full-depth baseline in accuracy (97.0% vs. 97.3%), recall (96.9% vs. 97.1%), and F1-score (97.0% vs. 97.3%). At the same time, it reduced average latency from 70.4 ms to 46.1 ms, and nearly halved energy use per patch from 0.124 J to 0.070 J–enough to process a full Sentinel-2 scene (2401 patches) within a 5-minute downlink window. The model also remains highly compact, occupying just 7.3 MB–less than 8% of the 100 MB limit defined by the mission constraints. All key requirements–latency, compute, power, and memory–were met with comfortable margins, demonstrating that advanced visual inference can run directly aboard a small satellite.

Hardware experiments confirmed this potential, while also highlighting the importance of platform choice. On the Jetson Nano, dynamic branching allowed early exits to operate entirely on-device, producing the efficiency gains described above. But on the Intel Myriad X, which lacks graph-level control flow, the model had to be split into sub-models, each triggered by host logic. The resulting host–device transfers introduced overhead that cancelled out much of the benefit–sometimes even increasing latency and energy despite comparable accuracy. This contrast reinforces a key insight: early-exit architectures are only effective when supported natively by the inference engine and hardware.

Naturally, there are limitations. The use of only three spectral bands leaves some fire cues untapped–especially for smoldering or cloud-obscured fires. The dataset, while geographically diverse, spans only a limited time range, so seasonal variation may reduce accuracy over long missions. And the Jetson-class hardware assumes the satellite can dissipate ~3 W of thermal load–feasible, but nontrivial during eclipse.

Looking forward, several improvements are possible. Incorporating additional bands (e.g., SWIR-1) or derived indices (like NBR) could help detect low-temperature fires. Exit thresholds could be adjusted in real time using telemetry, trading certainty for power as needed. And porting the architecture to radiation-hardened FPGAs would enable conditional inference without relying on a general-purpose GPU.

In conclusion, this thesis confirms the central hypothesis: many Earth observation images can be confidently classified at shallow depths, and by leveraging that asymmetry, a small satellite can detect wildfires within strict limits on power, mass, and bandwidth. The proposed approach demonstrates that onboard wildfire detection is technically feasible and ready for integration into future CubeSat missions.

# References


[1] Our World in Data, “Annual number of objects launched into space.” Accessed: Mar. 16, 2025. [Online]. Available: https://ourworldindata.org/grapher/yearly-number-of-objects-launched-into-outer-space?country=~OWID_WRL

[2] Orbiting now, “Active satellite orbit data.” Accessed: Mar. 16, 2025. [Online]. Available: https://orbit.ing-now.com/

[3] C. Daehnick, J. Gang, and I. Rozenkopf, “Space launch: Are we heading for oversupply or a shortfall?” Accessed: Mar. 16, 2025. [Online]. Available: https://www.mckinsey.com/industries/aerospace-and-defense/our-insights/space-launch-are-we-heading-for-oversupply-or-a-shortfall

[4] Q. Li, S. Wang, X. Ma, A. Zhou, and F. Yang, “Towards Sustainable Satellite Edge Computing,” Mar. 2022, Accessed: Mar. 16, 2025. [Online]. Available: https://arxiv.org/abs/2203.06065

[5] J. Claricoats and S. M. Dakka, “Design of Power, Propulsion, and Thermal Sub-Systems for a 3U CubeSat Measuring Earth’s Radiation Imbalance,” *Aerospace*, vol. 5, no. 2, p. 63, Jun. 2018, doi: 10.3390/aerospace5020063.

[6] G. Giuffrida *et al.*, “The Φ-Sat-1 Mission: The First On-Board Deep Neural Network Demonstrator for Satellite Earth Observation,” *IEEE Transactions on Geoscience and Remote Sensing*, vol. 60, pp. 1–14, Nov. 2021, doi: 10.1109/TGRS.2021.3125567.

[7] Science and Technology, “Technology to Reduce the Impacts of Wildfires.” Accessed: Mar. 17, 2025. [Online]. Available: https://www.dhs.gov/science-and-technology/technology-reduce-impacts-wildfires

[8] A. Mohapatra and T. Trinh, “Early Wildfire Detection Technologies in Practice—A Review,” *Sustainability*, vol. 14, no. 19, p. 12270, Sep. 2022, doi: 10.3390/su141912270.

[9] Z. Hong *et al.*, “Active Fire Detection Using a Novel Convolutional Neural Network Based on Himawari-8 Satellite Images,” *Front Environ Sci*, vol. 10, Mar. 2022, doi: 10.3389/fenvs.2022.794028.

[10] A. Duggan, B. Andrade, and H. Afli, “Advancing Earth Observation: A Survey on AI-Powered Image Processing in Satellites,” Jan. 2025, Accessed: Mar. 19, 2025. [Online]. Available: https://arxiv.org/html/2501.12030v1

[11] S. Teerapittayanon, B. McDanel, and H. T. Kung, “BranchyNet: Fast Inference via Early Exiting from Deep Neural Networks,” Sep. 2017, Accessed: Mar. 19, 2025. [Online]. Available: https://arxiv.org/abs/1709.01686

[12] M. J. Wooster *et al.*, “Satellite remote sensing of active fires: History and current status, applications and future requirements,” *Remote Sens Environ*, vol. 267, Dec. 2021, doi: 10.1016/j.rse.2021.112694.

[13] D. Lobb, R. Bird, M. Wooster, and A. Regan, “Fire detection from LEO: trade-offs for selection of spectral bands and a wide-swath optical design using MWIR and visible bands,” in *International Conference on Space Optics — ICSO 2018*, N. Karafolas, Z. Sodnik, and B. Cugny, Eds., SPIE, Jul. 2019. doi: 10.1117/12.2536062.

[14] L. Giglio, W. Schroeder, and C. O. Justice, “The collection 6 MODIS active fire detection algorithm and fire products,” *Remote Sens Environ*, vol. 178, pp. 31–41, Jun. 2016, doi: 10.1016/j.rse.2016.02.054.
[15] NOAA SATELLITE AND INFORMATION SERVICE, “GOES-R Fire Detection and Characterization,” Mar. 25, 2025. Accessed: Mar. 25, 2025. [Online]. Available: https://www.goes-r.gov/education/docs/fs_fire.pdf
[16] JAXA, “Himawari-8 Wild Fire product.” Accessed: Mar. 25, 2025. [Online]. Available: https://www.eorc.jaxa.jp/ptree/documents/README_H08_L2WLF.txt
[17] C. C. Schmidt, J. Hoffman, E. Prins, and S. Lindstrom, “GOES-R Advanced Baseline Imager (ABI) Algorithm Theoretical Basis Document For Fire / Hot Spot Characterization,” Oct. 2020. Accessed: Mar. 27, 2025. [Online]. Available: https://www.star.nesdis.noaa.gov/atmospheric-composition-training/documents/ABI_FDC_ATBD.pdf
[18] G. Tylor, “How to Overcome Satellites Limitations in Early Wildfire Detection.” Accessed: Mar. 27, 2025. [Online]. Available: https://www.exci.ai/real-time-satellites-limitations/
[19] Z. Hong *et al.*, “Active Fire Detection Using a Novel Convolutional Neural Network Based on Himawari-8 Satellite Images,” *Front Environ Sci*, vol. 10, Mar. 2022, doi: 10.3389/fenvs.2022.794028.
[20] W. Xu, M. J. Wooster, J. He, and T. Zhang, “Improvements in high-temporal resolution active fire detection and FRP retrieval over the Americas using GOES-16 ABI with the geostationary Fire Thermal Anomaly (FTA) algorithm,” *Science of Remote Sensing*, vol. 3, Jun. 2021, doi: 10.1016/j.srs.2021.100016.
[21] N. Maeda and H. Tonooka, “Early Stage Forest Fire Detection from Himawari-8 AHI Images Using a Modified MOD14 Algorithm Combined with Machine Learning,” *Sensors*, vol. 23, no. 1, Dec. 2022, doi: 10.3390/s23010210.
[22] M. Berman *et al.*, “Quantifying burned area of wildfires in the western United States from polar-orbiting and geostationary satellite active-fire detections,” *Int J Wildland Fire*, Apr. 2023, Accessed: Apr. 02, 2025. [Online]. Available: https://www.researchgate.net/publication/370307953_Quantifying_burned_area_of_wildfires_in_the_western_United_States_from_polar-orbiting_and_geostationary_satellite_active-fire_detections
[23] M. Yebra, N. Barnes, C. Bryant, G. J. Cary, and S. Durrani, “An integrated system to protect Australia from catastrophic bushfires.” Accessed: Apr. 05, 2025. [Online]. Available: https://knowledge.aidr.org.au/resources/ajem-october-2021-an-integrated-system-to-protect-australia-from-catastrophic-bushfires
[24] OroraTech, “Planet Earth Gets a Thermometer: OroraTech Launches First Dedicated Satellite With SpaceX.” Accessed: Apr. 05, 2025. [Online]. Available: https://ororatech.com/resources/news-blog/ororatech-first-satellite-launch-press-release
[25] OroraTech, “OroraTech Launches World’s First Satellite Constellation for Wildfire Detection & Data Accumulation.” Accessed: Apr. 05, 2025. [Online]. Available: https://ororatech.com/resources/news-blog/ororatech-launches-world-s-first-satellite-constellation-for-wildfire-detection-and-data-accumulation
[26] J. Foust, “OroraTech raises 25 million euros for satellite wildfire detection system.” Accessed: May 05, 2025. [Online]. Available: https://spacenews.com/ororatech-raises-25-million-euros-for-satellite-wildfire-detection-system/

[27] Muon Space, “Muon Space and Earth Fire Alliance Unveil FireSat Constellation, a Revolutionary Space Mission to Transform Global Wildfire Response.” Accessed: Apr. 05, 2025. [Online]. Available: https://www.muonspace.com/press/muon-space-and-earth-fire-alliance-unveil-firesat-constellation-a-revolutionary-space-mission-to-transform-global-wildfire-response
[28] “The first FireSat satellite has launched to help detect smaller wildfires earlier.” Accessed: Apr. 06, 2025. [Online]. Available: https://blog.google/feed/firesat-first-satellite-launch/
[29] N. Longépé *et al.*, “ΦSAT-2 MISSION OVERVIEW FOR THE #ORBITALAI CHALLENGE,” Feb. 2023. Accessed: Apr. 06, 2025. [Online]. Available: https://ai4eo.eu/wp-content/uploads/2023/02/Phisat-2_Mission_Overview_Web.pdf
[30] N. Melega *et al.*, “Development and implementation of the Φsat-2 mission,” in *Small Satellites Systems and Services Symposium (4S 2024)*, M. Petrozzi-Ilstad, Ed., SPIE, Mar. 2025. doi: 10.1117/12.3062624.
[31] M. Azami, N. Orger, V. Schulz, T. Oshiro, and M. Cho, “Earth Observation Mission of a 6U CubeSat with a 5-Meter Resolution for Wildfire Image Classification Using Convolution Neural Network Approach,” *Remote Sens (Basel)*, vol. 14, no. 8, Apr. 2022, doi: 10.3390/rs14081874.
[32] In The Sky, “KITSUNE.” Accessed: Apr. 07, 2025. [Online]. Available: https://in-the-sky.org/spacecraft.php?id=52148
[33] G. Mateo-Garcia *et al.*, “In-orbit demonstration of a re-trainable machine learning payload for processing optical imagery,” *Sci Rep*, vol. 13, no. 1, p. 10391, Jun. 2023, doi: 10.1038/s41598-023-34436-w.
[34] P. Barmpoutis, P. Papaioannou, K. Dimitropoulos, and N. Grammalidis, “A Review on Early Forest Fire Detection Systems Using Optical Remote Sensing,” *Sensors*, vol. 20, no. 22, p. 6442, Nov. 2020, doi: 10.3390/s20226442.
[35] A. G. Howard *et al.*, “MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications,” Apr. 2017, Accessed: Apr. 10, 2025. [Online]. Available: https://arxiv.org/abs/1704.04861
[36] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, “MobileNetV2: Inverted Residuals and Linear Bottlenecks,” Jan. 2018, Accessed: Apr. 10, 2025. [Online]. Available: https://arxiv.org/abs/1801.04381
[37] J. Manning, D. Langerman, B. Ramesh, E. Gretok, C. Wilson, and A. George, “Machine-Learning Space Applications on SmallSat Platforms with TensorFlow,” Jul. 2018, Accessed: Apr. 10, 2025. [Online]. Available: https://digitalcommons.usu.edu/smallsat/2018/all2018/458/
[38] M. Tan and Q. V. Le, “EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks,” May 2019, Accessed: Apr. 10, 2025. [Online]. Available: https://arxiv.org/abs/1905.11946
[39] N. Schizas, A. Karras, C. Karras, and S. Sioutas, “TinyML for Ultra-Low Power AI and Large Scale IoT Deployments: A Systematic Review,” *Future Internet*, vol. 14, no. 12, p. 363, Dec. 2022, doi: 10.3390/fi14120363.
[40] Y. Li *et al.*, “Implementation of Multi-Exit Neural-Network Inferences for an Image-Based Sensing System with Energy Harvesting,” *Journal of Low Power Electronics and Applications*, vol. 11, no. 3, p. 34, Sep. 2021, doi: 10.3390/jlpea11030034.
[41] L. Zeng, E. Li, Z. Zhou, and X. Chen, “Boomerang: On-Demand Cooperative Deep Neural Network Inference for Edge Intelligence on the Industrial Internet

of Things," *IEEE Netw*, vol. 33, no. 5, pp. 96–103, Sep. 2019, doi: 10.1109/MNET.001.1800506.
[42] A. Ehteshami Bejnordi and R. Krestel, "Dynamic Channel and Layer Gating in Convolutional Neural Networks", Accessed: Apr. 12, 2025. [Online]. Available: https://hpi.de/fileadmin/user_upload/fachgebiete/naumann/publications/PDFs/2020_bejnordi_dynamic.pdf
[43] N. Solanki, S. Tabrizchi, S. Sohrabi, J. Schmidt, and A. Roohi, "ATM-Net: Adaptive Termination and Multi-Precision Neural Networks for Energy-Harvested Edge Intelligence," Feb. 2025, Accessed: Apr. 12, 2025. [Online]. Available: https://arxiv.org/abs/2502.09822
[44] Y. Wu, Z. Wang, Z. Jia, Y. Shi, and J. Hu, "Intermittent Inference with Nonuniformly Compressed Multi-Exit Neural Network for Energy Harvesting Powered Devices," Apr. 2020, Accessed: Apr. 12, 2025. [Online]. Available: https://arxiv.org/abs/2004.11293
[45] C. P. S. The CubeSat Program, "CubeSat Design Specification," Feb. 2022. Accessed: Apr. 13, 2025. [Online]. Available: https://static1.squarespace.com/static/5418c831e4b0fa4ecac1bacd/t/62193b7fc9e72e0053f00910/1645820809779
[46] ISISPACE, "Small Satellite Solar Panels." Accessed: Mar. 14, 2025. [Online]. Available: https://www.isispace.nl/product/isis-cubesat-solar-panels/
[47] R. Hamann, J. Bouwmeester, and G. Brouwer, "Delfi-C3 Preliminary Mission Results," Feb. 2013, Accessed: Mar. 13, 2025. [Online]. Available: https://digitalcommons.usu.edu/smallsat/2009/all2009/25/
[48] C. Clark, "Huge Power Demand...Itsy-Bitsy Satellite: Solving the CubeSat Power Paradox," Feb. 2013, Accessed: Mar. 14, 2025. [Online]. Available: https://digitalcommons.usu.edu/smallsat/2010/all2010/19/
[49] SatCatalog, "6U Solar Panel." Accessed: Mar. 14, 2025. [Online]. Available: https://www.satcatalog.com/component/6u-solar-panel/
[50] Cubesat Market, "DMSA 6U/12U: Multifunction Solar Array with embedded antennas and sensors." Accessed: Apr. 14, 2025. [Online]. Available: https://www.cubesat.market/dmsa-6u-deployable-multifunction-solar-array
[51] V. Knap, L. K. Vestergaard, and D.-I. Stroe, "A Review of Battery Technology in CubeSats and Small Satellite Solutions," *Energies (Basel)*, vol. 13, no. 16, p. 4097, Aug. 2020, doi: 10.3390/en13164097.
[52] SatNow, "6U small satellite platform." Accessed: Apr. 12, 2025. [Online]. Available: https://www.satnow.com/products/satellite-bus-platforms/open-cosmos/45-1253-6u-small-satellite-platform
[53] J. A. Ruiz-de-Azua *et al.*, "6GStarLab -- A CubeSat Mission to support the development and standardization of Non-Terrestrial Networks towards 6G," Mar. 2025, Accessed: Apr. 09, 2025. [Online]. Available: https://www.researchgate.net/publication/390019083_6GStarLab_--_A_CubeSat_Mission_to_support_the_development_and_standardization_of_Non-Terrestrial_Networks_towards_6G
[54] M. H. Bin Azami *et al.*, "Design and environmental testing of imaging payload for a 6 U CubeSat at low Earth orbit: KITSUNE mission," *Frontiers in Space Technologies*, vol. 3, Nov. 2022, doi: 10.3389/frspt.2022.1000219.
[55] Intel Company, "Enhanced Visual Intelligence at the Network Edge." Accessed: Apr. 16, 2025. [Online]. Available: https://www.intel.com/content/dam/www/public/us/en/documents/product-briefs/myriad-x-product-brief.pdf

[56] N. Oh, “Intel Announces Movidius Myriad X VPU, Featuring ‘Neural Compute Engine.’” Accessed: Apr. 16, 2025. [Online]. Available: https://www.anandtech.com/show/11771/intel-announces-movidius-myriad-x-vpu
[57] E. R. Dunkel *et al.*, “Benchmarking Deep Learning Models on Myriad and Snapdragon Processors for Space Applications.” Accessed: Apr. 16, 2025. [Online]. Available: https://ai.jpl.nasa.gov/public/documents/papers/dunkel-jais-benchmarking23.pdf
[58] “Edge TPU performance benchmarks.” Accessed: Apr. 15, 2025. [Online]. Available: https://coral.ai/docs/edgetpu/benchmarks/
[59] G. Lentaris *et al.*, “Performance and Radiation Testing of the Coral TPU Co-processor for AI Onboard Satellites,” in *2023 European Data Handling & Data Processing Conference (EDHPC)*, IEEE, Oct. 2023, pp. 1–4. doi: 10.23919/EDHPC59100.2023.10396640.
[60] “NVIDIA Jetson Nano.” Accessed: Apr. 15, 2025. [Online]. Available: https://www.nvidia.com/en-eu/autonomous-machines/embedded-systems/jetson-nano/product-development/
[61] A. Rodríguez-Molina *et al.*, “Bentayga-I: Development of a Low-Cost and Open-Source Multispectral CubeSat for Marine Environment Monitoring and Prevention,” *Sensors*, vol. 24, no. 23, p. 7648, Nov. 2024, doi: 10.3390/s24237648.
[62] “NVIDIA Jetson Xavier.” Accessed: Apr. 17, 2025. [Online]. Available: https://www.nvidia.com/en-eu/autonomous-machines/embedded-systems/jetson-xavier-series/
[63] D. Saaristo, “Shoot From the Stars: Startup Provides Early Detection of Wildfires From Space.” Accessed: Apr. 17, 2025. [Online]. Available: https://blogs.nvidia.com/blog/ororatech-wildfires-from-space/
[64] “Jetson Modules.” Accessed: Apr. 15, 2025. [Online]. Available: https://developer.nvidia.com/embedded/jetson-modules
[65] R. Pitonak, J. Mucha, L. Dobis, M. Javorka, and M. Marusin, “CloudSatNet-1: FPGA-Based Hardware-Accelerated Quantized CNN for Satellite On-Board Cloud Coverage Classification,” *Remote Sens (Basel)*, vol. 14, no. 13, p. 3180, Jul. 2022, doi: 10.3390/rs14133180.
[66] A. C. McCormick, “Space Tolerant CNN FPGA Deployment, Part 3”, Accessed: Apr. 17, 2025. [Online]. Available: https://alpha-data.com/pdfs/ad-an-0118_v1_0.pdf
[67] Y. Xu, J. Luo, and W. Sun, “Flare: An FPGA-Based Full Precision Low Power CNN Accelerator with Reconfigurable Structure,” *Sensors*, vol. 24, no. 7, p. 2239, Mar. 2024, doi: 10.3390/s24072239.
[68] “Overview of TI Deep Learning Library.” Accessed: Apr. 18, 2025. [Online]. Available: https://software-dl.ti.com/jacinto7/esd/processor-sdk-rtos-jacinto7/06_02_00_21/exports/docs/tidl_j7_01_01_00_10/ti_dl/docs/user_guide_html/md_tidl_overview.html
[69] “Fire Information for Resource Management System.” Accessed: Apr. 20, 2025. [Online]. Available: https://firms.modaps.eosdis.nasa.gov/
[70] “A planetary-scale platform for Earth science data & analysis.” Accessed: Apr. 20, 2025. [Online]. Available: https://earthengine.google.com/
[71] Earth Engine Data Catalog, “Harmonized Sentinel-2 MSI: MultiSpectral Instrument, Level-2A (SR).” Accessed: Apr. 20, 2025. [Online]. Available:

https://developers.google.com/earth-engine/datasets/catalog/COPERNICUS_S2_SR_HARMONIZED#description
[72] H. Singh, L.-M. Ang, and S. K. Srivastava, “Active wildfire detection via satellite imagery and machine learning: an empirical investigation of Australian wildfires,” *Natural Hazards*, Mar. 2025, doi: 10.1007/s11069-025-07163-w.
[73] N. Scott, “Z-Score: Meaning and Formula.” Accessed: Apr. 22, 2025. [Online]. Available: https://www.investopedia.com/terms/z/zscore.asp
[74] “Albumentations.” Accessed: Apr. 22, 2025. [Online]. Available: https://pypi.org/project/albumentations/1.3.0/
[75] “Wildfire dataset.” Accessed: May 12, 2025. [Online]. Available: https://drive.google.com/drive/folders/1vB2_x5n1sWb7kOiMR3Oy2mNmi1FAqxaH?usp=drive_link
[76] K. He, X. Zhang, S. Ren, and J. Sun, “Deep Residual Learning for Image Recognition,” Dec. 2015, Accessed: Apr. 23, 2025. [Online]. Available: https://arxiv.org/abs/1512.03385
[77] “MobileNet, MobileNetV2, and MobileNetV3.” Accessed: Apr. 15, 2025. [Online]. Available: https://keras.io/api/applications/mobilenet/
[78] “CrossEntropyLoss.” Accessed: Apr. 22, 2025. [Online]. Available: https://docs.pytorch.org/docs/stable/generated/torch.nn.CrossEntropyLoss.html
[79] A. Bosler, “EarlyExitCNN repository.” Accessed: May 12, 2025. [Online]. Available: https://github.com/aleksandr-bosler/EarlyExitCNN

# Appendix 1 – Non-exclusive licence for reproduction and publication of a graduation thesis[1]

I Aleksandr Bosler

1. Grant Tallinn University of Technology free licence (non-exclusive licence) for my thesis “Adaptive Energy-Efficient CNN for Onboard Wildfire Detection on CubeSats”, supervised by Aleksei Tepljakov
   1.1. to be reproduced for the purposes of preservation and electronic publication of the graduation thesis, incl. to be entered in the digital collection of the library of Tallinn University of Technology until expiry of the term of copyright;
   1.2. to be published via the web of Tallinn University of Technology, incl. to be entered in the digital collection of the library of Tallinn University of Technology until expiry of the term of copyright.
2. I am aware that the author also retains the rights specified in clause 1 of the non-exclusive licence.
3. I confirm that granting the non-exclusive licence does not infringe other persons' intellectual property rights, the rights arising from the Personal Data Protection Act or rights arising from other legislation.

12.05.2025

1 The non-exclusive licence is not valid during the validity of access restriction indicated in the student's application for restriction on access to the graduation thesis that has been signed by the school's dean, except in case of the university's right to reproduce the thesis for preservation purposes only. If a graduation thesis is based on the joint creative activity of two or more persons and the co-author(s) has/have not granted, by the set deadline, the student defending his/her graduation thesis consent to reproduce and publish the graduation thesis in compliance with clauses 1.1 and 1.2 of the non-exclusive licence, the non-exclusive license shall not be valid for the period.

# Appendix 2 – Use of Generative Artificial Intelligence

Generative artificial intelligence, specifically ChatGPT, was used in the preparation of this thesis for translation, improving readability, and enhancing clarity of expression.