POLITECNICO DI TORINO
Repository ISTITUZIONALE

A multi-service edge-AI architecture based on self-supervised learning

Original
A multi-service edge-AI architecture based on self-supervised learning / Magli, E., Angarano, S., Bassetti, S., Bianchi, T., Boccardo, P., Bucci, S., Chiaberge, M., Inzerillo, G., Lisi, D., Mascetti, G., Mergè, M., Monaco, C., Pasturensi, M., Piccinini, D., Valsesia, D., Zema, G.. - ELETTRONICO. - (2024), pp. 1-9. (75th International Astronautical Congress Milan (Ita) 14-18 October 2024).

Availability:
This version is available at: 11583/2995769 since: 2025-01-10T16:09:21Z

Publisher:
IAF

Published
DOI:

Terms of use:

This article is made available under terms and conditions as specified in the corresponding bibliographic description in the repository

Publisher copyright

(Article begins on next page)

26 June 2026

75th International Astronautical Congress (IAC), Milan, Italy, 14-18 October 2024.
Copyright ©2024 by the International Astronautical Federation (IAF). All rights reserved.

IAC-24-B1.4.2

A multi-service edge-AI architecture based on self-supervised learning

E. Magli[a]*, S. Angarano[a], S. Bassetti[b], T. Bianchi[a], P. Boccardo[a], S. Bucci[c], M. Chiaberge[a], G. Inzerillo[a], D. Lisi[b], G. Mascetti[d], M. Mergè[d], C. Monaco[b], M. Pasturensi[c], D. Piccinini[a], D. Valsesia[a], G. Zema[c]

[a] *Department of Department of Electronics and Telecommunications, Politecnico di Torino, Corso Duca degli Abruzzi 24, Torino, Italy.*
[b] *Ithaca srl, Via Pier Carlo Boggio 61, Torino, Italy.*
[c] *Argotec srl, Via Cervino 52, Torino, Italy.*
[d] *Agenzia Spaziale Italiana, Via del Politecnico snc, Roma, Italy.*
* Corresponding Author

Abstract

This paper presents the development of an efficient artificial intelligence (AI) methodologies for onboard satellite image processing. The proposed system features a fast and efficient neural network designed to process radiometrically corrected multispectral optical images directly onboard satellites. The architecture is composed of a backbone feature extractor that generates semantically meaningful feature representations of input data, which are shared with multiple task-specific heads for various applications, including image classification, segmentation, and object detection. Training employs a self-supervised learning approach, significantly reducing the need for labelled data, with only small application-specific datasets required. The flexible design allows new tasks to be added without retraining the entire model or making major code changes. To ensure suitability for onboard use, the model is optimized for efficiency and low energy consumption through the use of quantization techniques and efficient deep learning modules. Key applications include cloud segmentation, fire detection, and flood detection, which demand low-latency responses for early warning and damage assessment.

Keywords: Artificial Intelligence, Self-Supervised Learning, Multitask learning, Onboard Edge-device, Remote Sensing

Acronyms/Abbreviations

AI Artificial Intelligence
CNN Convolutional Neural Network
DL Deep Learning
FP Floating-Point precision
INT Integer precision
IoU Intersection over Union
NLL Negative Log Likelihood
SL Supervised Learning
SSL Self-Supervised Learning

1. Introduction

Conventional satellite imaging systems typically involve capturing data in space and then transmitting it to ground stations for subsequent processing. This workflow often results in significant delays, sometimes extending to several days, before the final processed images are made available to users. Such delays are especially problematic in urgent scenarios, such as natural disasters, where immediate access to data is critical. To mitigate this issue, a new approach has gained traction: shifting some of the image processing workload onboard the satellite itself. By processing data in real time, satellites can detect critical events early and send prioritized alerts, ensuring that essential information reaches ground stations faster.

This evolving trend, known as Edge-AI, capitalizes on recent advancements in artificial intelligence, particularly deep learning, to enable advanced image analysis directly in orbit. Edge-AI allows satellites to process imagery immediately after it is captured, filtering out unimportant data, such as cloud-covered images, and detecting high-priority events, like floods or fires, with minimal delay. However, the adoption of Edge-AI has been hindered by several challenges, including the difficulty of deploying machine learning models on field-programmable gate arrays (FPGAs), high power consumption, and the limited availability of labelled data for training purposes.

The objective of this paper is to establish a framework for developing an Edge-AI system that can manage multiple onboard tasks while addressing the existing technological barriers. The proposed system revolves around a deep neural network specifically designed for satellites carrying multispectral sensors. The central component of this system is a feature extractor, which generates semantic representations of the captured multispectral imagery that can be utilized by various onboard applications. This extractor is trained using a self-supervised learning (SSL) technique, enabling it to generate general-purpose features without requiring large quantities of labelled data. Individual application-specific heads are developed for specific tasks and utilize

Copyright ©2024 by the International Astronautical Federation (IAF). All rights reserved.

these shared features, minimizing both computational demands and the need for extensive annotated datasets. Additionally, multitasking can improve overall system efficiency; for example, cloud detection can be used to optimize data transmission by excluding regions with many clouds in order to adjust compression algorithms for better data management and a faster data transmission.

While deep learning significantly enhances detection capabilities in such applications, designing an AI system for onboard real-time multitasking requires careful consideration of satellite platform constraints, including limited power and processing resources. This paper proposes the development of such an architecture, emphasizing a lightweight neural network architecture tailored for feature extraction from multispectral images with varying spatial resolutions. The feature extractor is trained in a self-supervised manner using large volumes of unlabelled data. These extracted features are made available to third-party developers, who can design task-specific application heads, such as for image segmentation or classification, each running independently of the backbone architecture. This ensures that multiple applications can operate concurrently without overwhelming the onboard computational resources.

Furthermore, the modular design of the system supports in-flight updates and the gradual integration of new tasks, thereby improving the satellite's adaptability and long-term efficiency. This paper presents an implementation of the whole architecture, comprising the feature extractor and three specific application heads, with results demonstrating its effectiveness in three different tasks: cloud segmentation, flood segmentation and fire segmentation, highlighting the potential of this architecture for onboard satellite image processing. To perform the latter tasks, we selected five spectral bands for trainings, Red, Green, Blue, Near-Infrared (NIR), and Short-Wave Infrared (SWIR), as they offer a strong balance between informational richness and computational efficiency. These bands provide sufficient spectral diversity to capture key features for our tasks while maintaining efficiency, as we only use five channels rather than the dozens or hundreds typically found in hyperspectral imagery.

1.1 Onboard AI

The integration of artificial intelligence (AI) directly on-board satellites is deeply changing satellite observation of the Earth, enabling real-time data processing and decision-making capabilities. This marks a significant departure from the traditional model where data is transmitted to ground stations for post-processing, resulting in delays that can be harmful in critical situations such as natural disasters or environmental monitoring. However, while on board artificial intelligence has great potential, it presents a number of technical challenges. Zhang et al. [1] outline several key challenges, including the difficulty of efficiently integrating AI models with onboard hardware, managing the power consumption of these systems.

Benchmarking studies, such as the work by Ziaja et al. [2], have further underscored the difficulties in deploying deep learning models on space-qualified hardware. Their study systematically evaluated the performance of various neural network architectures on onboard space platforms, evaluating the trade-offs between model complexity and onboard resource constraints.

A key milestone in this field was the ESA ϕ-sat-1 mission**,** which demonstrated the feasibility of using deep learning algorithms for onboard processing as shown in [3]. In this mission, a convolutional neural network was deployed to filter out cloud-covered images directly in space, marking one of the first successful implementations of a deep neural network on a satellite platform.

1.2 Deep Learning models

Deep learning has emerged as a powerful technology in image processing and computer vision, enabling significant advances in tasks such as object detection, image classification, segmentation and more. Unlike traditional machine learning techniques, which often rely on hand-created features and domain-specific knowledge, deep learning models automatically learn hierarchical representations of data directly from raw inputs, making them highly effective in visual tasks.

At the core of deep learning's success in image processing are convolutional neural networks (CNNs), which have become the standard architecture for many computer vision tasks. CNNs are designed to exploit the spatial structure of images, using convolutional layers to capture patterns such as edges, textures, and higher-level features. This architecture has proven extremely effective in large-scale datasets, such as ImageNet, where deep learning models have consistently surpassed traditional methods in classification [4], segmentation [5] and object detection [6] tasks, just to name a few. This breakthrough ignited widespread adoption of CNNs in various domains, from medical imaging to remote sensing, where accurate and efficient image analysis is fundamental.

1.3 Self-Supervised Learning

Self-supervised learning (SSL) is a paradigm in machine learning where models learn to predict highly informative representation of the data from the images without the need of labelled ground truth, effectively leveraging unlabelled data for training. Unlike traditional supervised learning (SL), which requires labelled examples for each training instance, SSL utilizes the inherent structure in the data itself to learn rich features.

Copyright ©2024 by the International Astronautical Federation (IAF). All rights reserved.

This approach has proven especially useful in domains where labelled data is scarce or expensive to obtain, such as in image processing and computer vision.

Among the various self-supervised paradigms, one of the most successful is contrastive learning. Contrastive learning [7, 8] is a self-supervised learning technique that focuses on learning representations by distinguishing between similar and dissimilar data points. The core idea is to train models to bring similar examples (positive pairs) closer together in the feature space while pushing dissimilar examples (negative pairs) farther apart.

2. Material and methods

In this section we present a general overview of the proposed AI system, along with the design of the feature extractor and the application-specific heads. Furthermore, we present the dataset we specifically created and used for the trainings of both feature extractor and application heads.

2.1 Datasets

To train our AI architecture, we utilized four different datasets, all containing multispectral images captured by the European Space Agency's Sentinel-2 mission:

- A subset of approximately 40,000 images from the majorTom dataset [9] was used for self-supervised learning to train the backbone feature extractor.
- The CloudSen12 [10] was employed for supervised training of the application head dedicated to cloud segmentation. This dataset contains around 49.000 images with corresponding segmentation masks indicating the presence of cloudy or clear pixels (distinguishing between “thin cloud”, “thick clouds”, “cloud shadow” and “background” classes) and was specifically created as an extensive benchmark for cloud cover segmentation tasks.
- The datasets for trainings of the application heads for flood and fire segmentation were specifically created by us for these two tasks. For the flood segmentation dataset, for each image, we created segmentation masks with four possible values: “no event”, “floods”, “flood trace”, and “permanent water”. Similarly, for the fire segmentation dataset, we generated segmentation masks for each image with three possible values: “no event”, “burnt area”, and “active flame”. To generate the datasets, shapefiles obtained from the Rapid Mapping activations of the Copernicus Emergency Management Service were also used; these shapefiles were refined automatically and manually to meet our needs. [CEMS]

2.2 AI architecture

The overall architecture of our AI system is illustrated in Figure 1 and consists of two primary components: a backbone feature extractor and multiple application heads. This modular design enables the use of various independent application heads for different tasks, with the flexibility to add more as needed. The backbone serves as a universal feature extractor, trained using a self-supervised learning approach and developed independently of specific application heads. Its primary requirement is to generate features that are broadly applicable across a range of tasks. Once extracted, these features are passed to task-specific heads, smaller neural networks tailored to individual tasks and then fine-tuned on datasets that are specific for the task they are designed to solve. To maintain the reusability of the extracted features for all tasks, application heads are restricted from fine-tuning the backbone itself.

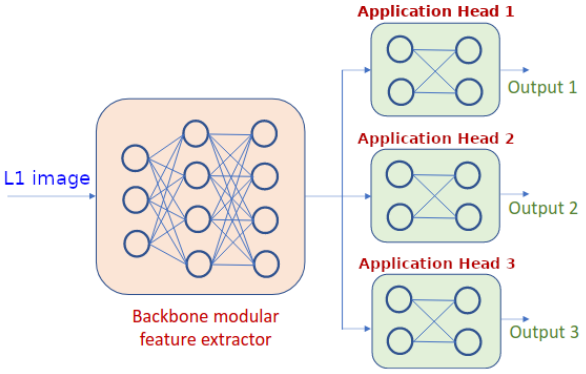

Fig. 1. The backbone feature extractor creates meaningful feature representations that are shared with the three application heads.

The input to the architecture, as shown in Figure 1, is a raw Sentinel-2’s Level-1C image received directly from the multispectral sensor. These images have undergone some radiometric and geometric corrections, and an orthorectification process has been applied. It is important to note that once the backbone has extracted the features, they are simultaneously passed to the application heads, which process these representations in parallel. This allows the system to perform the *n* (three, for our specific demonstration) tasks concurrently, effectively reducing latency.

Moreover, to meet the stringent requirements of low latency and computational efficiency for effective onboard processing, we applied a post-training quantization process, reducing the model's weight precision from 32-bit floating point (FP) to 8-bit integer (INT). This approach minimizes the model's complexity, reducing the memory occupation, with only a minimal loss in accuracy, optimizing it for deployment on resource-constrained hardware. The shift to INT also

Copyright ©2024 by the International Astronautical Federation (IAF). All rights reserved.

allows for faster inference times, making the model suitable for real-time operations in space environments.

In the following subsections we are now going into details and explain each module of our architecture.

2.2.1 Backbone Feature Extractor and SSL Training

As mentioned in Section 2, the backbone feature extractor has the goal of generating feature representations that must be shared among all application heads. Sharing the feature extractor across multiple information extraction applications is a highly effective concept for enabling the execution of several tasks within the same architecture. The most computationally intensive part is performed only once, significantly improving efficiency. To achieve this, the feature extractor must be trained independently of any specific application, as it needs to be adaptable to a wide range of tasks. For this purpose, we trained the feature extractor using self-supervised learning techniques, and more specifically on contrastive learning. In contrast to conventional supervised training, self-supervised techniques do not rely on explicit annotations and labels from the training dataset.

To perform the self-supervised training of our feature extractor, we employed the contrastive learning method BYOL [8]. Contrastive learning is a self-supervised approach that aims to learn useful representations by contrasting positive pairs (i.e., similar data points, such as different views of the same image) against negative pairs (i.e., dissimilar data points). Typically, these architectures utilize two neural networks, an online network and a target network, to align their output representations effectively.

The architecture of our backbone and the training process using the SSL framework are illustrated in Figure 2. As depicted in the figure, the input to the SSL architecture consists of two distinct augmentations of the same base image. These augmentations play a crucial role, as they introduce variations of the same input that both the online and target networks process. Despite receiving different augmented versions, the networks are trained to produce similar feature representations. Specifically, for the online network (upper branch), the input image is subjected solely to spatial augmentations (e.g., random crops, horizontal and vertical flips). In contrast, the target network (lower branch) receives the same source image augmented with both spatial and color transformations (e.g., solarization, color jittering).

Then, in order to learn informative feature representations of the input images, a global contrastive loss is computed by aligning the feature representations output by the global projector and global predictor of the online network with those from the target network. The goal is to minimize the difference between the predicted output from the online network and the projected output from the target network.

It is important to highlight that most state-of-the-art SSL methods, including BYOL, are primarily designed for image-level tasks, such as classification. However, in this work, we aim to pre-train a model that can effectively support a variety of application heads, including those responsible for pixel-level tasks like semantic segmentation and object detection. To achieve this, we introduced a Local Contrastive Loss inspired by [11] into our SSL training. This loss is specifically designed to better capture, and model, dense features, which are crucial in solving fine-grained tasks.

The overall SSL loss is computed as:

$$L = \lambda L_{LC} + (1 - \lambda) L_{GC}$$

Where λ is a trade-off parameter to balance between local contrastive loss (L_{LC}) and the classical global contrastive loss (L_{GC}).

The local contrastive loss is essentially a Negative Log-Likelihood (NLL) to capture fine-grained, pixel-level correspondences between two differently augmented views of the same input image. To do so, first, we define a set of key points in one augmented image and then map these points to their corresponding locations in the second image based on the known spatial transformations. These key points in the first image (i.e., p_c) are matched with corresponding points in the second image (i.e., p_{sc}). For each key point in the first image, the model computes the similarity between its feature representation and the feature representation of the corresponding point in the second image. This is done using a dense correspondence map, where the similarity score between points is computed. Lastly, The NLL quantifies how likely the corresponding points (i.e., p_c and p_{sc}) are to match in terms of their feature similarity. Specifically, it measures how well the feature representation of a point in the first image aligns with the feature representation of the corresponding point in the second image, compared to other possible points in the second image.

On the other hand, the global contrastive loss uses cosine similarity to measure how well the global representations of the two augmented views align. We apply the cosine similarity to the two features (tensors) outputted from the online and target network.

Regarding the architecture of the backbone feature extractor, given that the objective is to develop a model capable of running within the constraints of a satellite payload, we opted for a pre-existing SOTA efficient and low-complexity architecture [12]. Nevertheless, it is worth noting that the choice of this model as the backbone is not restrictive, as the SSL training procedure, as well as the entire framework, would remain unchanged with any other neural network capable of extracting image features.

Copyright ©2024 by the International Astronautical Federation (IAF). All rights reserved.

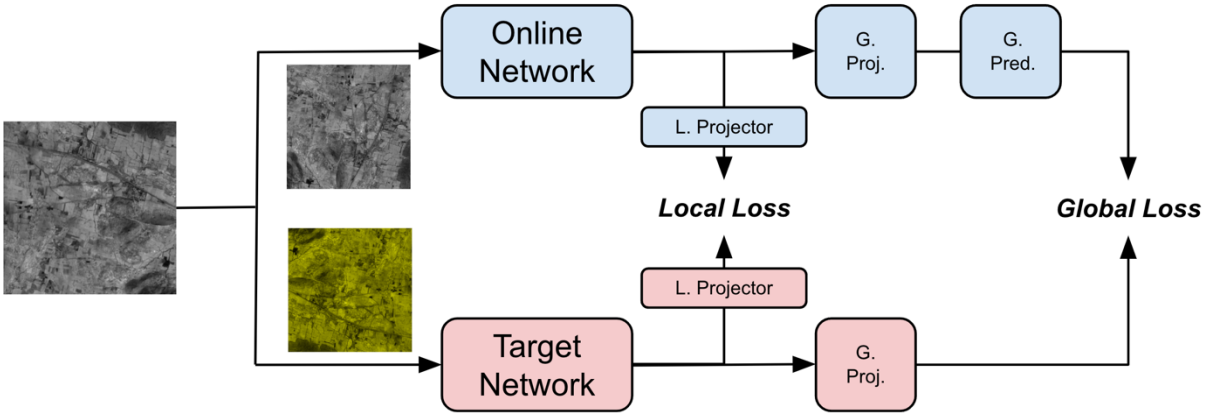

Fig. 2. SSL training of the backbone feature extractor using both global and local contrastive loss. The input images are obtained by applying to the same source image two different sets of augmentations: spatial transformations (upper branch) and spatial and color transformations (lower branch).

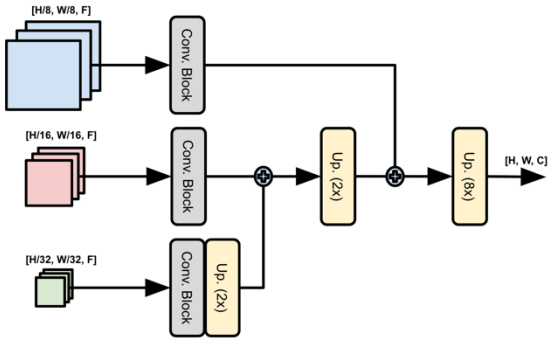

Fig. 3. Architecture of the segmentation head which takes in input multi-resolution features. The shape of the different features is marked in squared brackets; F identify the number of channels computed by the feature extractor, C the number of output classes in the segmentation map, H and W are the height and width of the image/features, respectively.

2.2.2 Application Heads

To evaluate our AI system, we tested the entire model on the three tasks described in Section 2 ("Materials and Methods"). Consequently, we required three parallel, task-specific heads. Since all the tasks are essentially segmentation tasks (clouds, fires, and floods), we designed a single application head specifically for segmentation. As is shown in Figure 3, the segmentation head is a neural network that takes in input three different features maps, with different resolution, extracted by the feature extractor, and aggregate them before performing a set of upsampling and lightweight convolution to finally obtain the segmentation map.

Each head, tailored to specific tasks, only requires fine-tuning on datasets pertinent to its particular function. When introducing a new task, the process involves simply creating and fine-tuning an additional application head. Importantly, training the entire architecture, including both the backbone and the application head, is avoided to maintain the modularity and generality of the architecture. Instead, by selectively fine-tuning only the application heads while keeping the backbone's weights frozen (i.e., unchanged, without any fine-tuning on any specific dataset), the system's modularity is maintained, allowing for simultaneous inference of multiple tasks by different heads.

3. Results

We conducted a series of extensive experiments with the proposed architecture. The tests were performed by first training the backbone using the SSL approach explained in Subsection 2.2.1 ("Backbone Feature Extractor and SSL Training"). Then, the three segmentation heads for clouds, fires and floods segmentation have been trained, maintaining the feature extractor's weights frozen (i.e., without updating them), in a classical supervised way, thus using the labels of the three datasets.

Table 1 shows a summary of the computational complexity of each component of the proposed architecture, computed by forwarding an input image of shape $512 \times 512 \times 5$. It is easily noticed that the bottleneck of our architecture is found, as expected, in the backbone feature extractor, where the computational complexity is most concentrated, while the segmentation head design is significantly more efficient, with a small number of floating-point operations (FLOPs) and parameters compared to the backbone.

Copyright ©2024 by the International Astronautical Federation (IAF). All rights reserved.

To evaluate the performance of our model across the three tasks, cloud segmentation, fire segmentation, and flood segmentation, we adopted the Mean Intersection over Union (mIoU) metric. Since all of these tasks are fundamentally segmentation problems, mIoU is particularly well-suited as it is one of the most widely used metrics for assessing the quality of predicted segmentations in image analysis. The mIoU is calculated by first determining the Intersection over Union (IoU) for each class in the segmentation task. The IoU measures the overlap between the predicted segmentation and the ground truth for each class, where the intersection is the common area between the predicted and ground truth masks, and the union is the total area covered by both the predicted and ground truth masks combined. The mIoU extends this concept by averaging the IoU values across all the classes, providing a single performance measure.

Mathematically, for N classes, the mIoU is expressed as:

$$mIoU = \frac{1}{N}\sum_{i=1}^{N} \omega_i \cdot \left|\frac{P_i \cap G_i}{P_i \cup G_i}\right|$$

Where P_i is the predicted set of classes for class i, G_i is the ground truth set of pixels for class i, and ω_i is the weight assigned to class i, based on the proportion of pixels in the ground truth for that class.

Table 1. Computational complexity of each component of the proposed Architecture.

	FLOPs	Params
Backbone Feature Extractor	$4.50 \cdot 10^9$	$9.55 \cdot 10^6$
Segmentation Head	$187 \cdot 10^6$	$402 \cdot 10^3$

The mIoU metric is particularly advantageous in our case since it balances the contributions of all classes in the segmentation task, ensuring that performance is not disproportionately affected by classes that occupy larger areas (e.g., background), furthermore, using the weight factor ω ensures that larger classes (in terms of pixels) have a proportionally higher weight in the final mIoU calculation. By using mIoU as our evaluation metric, we can comprehensively assess how well our model performs across different segmentation tasks and ensure a fair comparison of its efficacy on cloud, fire, and flood segmentation tasks. This metric also allows for easy interpretability: a value closer to 1 indicates a higher degree of overlap between the predicted and true segmentation, whereas a value closer to 0 indicates poor overlap, making it a clear indicator of model performance.

In the following subsections we discuss in details the evaluation setting and the results obtained for each of the three tasks. As a term of comparison, we evaluate or architecture under two configurations:

- ***SSL*:** the backbone feature extractor is pretrained using the aforementioned SSL approach, after which its weights are frozen. The three task-specific heads are then fine-tuned independently for each task. This configuration allows for parallel inference across all three heads.
- ***SL*:** the backbone feature extractor and a single segmentation head are treated as one single network and trained using a classical supervised learning (SL) approach. In this setup, parallel inference is not possible; only one task can be addressed at a time. However, the backbone is fine-tuned specifically for that task, producing task-specific features rather than general features shared across multiple tasks.

3.1. Clouds Segmentation

To evaluate our architecture on the cloud segmentation task, we utilized only the high-quality images (i.e., images with a corresponding high-quality segmentation mask as ground truth) from the CloudSen12 dataset [10], specifically those that were manually labelled. We excluded the low-quality images, as they are either automatically annotated or contain labelling inaccuracies, which could introduce noise and negatively impact the model's training and evaluation process. By focusing on the high-quality subset, we ensured more reliable ground truth data, allowing for a more accurate assessment of the model's segmentation performance.

Moreover, since there is essentially no difference between “thin clouds” and “thick clouds” classes, we aggregated the two classes in one, resulting in a 3-classes dataset (“Clouds”, “Clouds Shadow”, “Background”).

Table 2 presents quantitative performance comparisons for the cloud segmentation task using both the SSL and SL configurations. As anticipated, the SL architecture exhibits superior performance since both the backbone feature extractor and the segmentation head are specifically trained and fine-tuned for the cloud segmentation task. However, this highest performance comes at a cost: it precludes the ability to perform other tasks in parallel, strongly increasing latency in the multitask approach. In Figure 4 (first column) are reported the confusion matrices computed for the cloud segmentation task.

For cloud segmentation, the SSL model performs reasonably well in detecting the background (47.48%), but shows notable confusion between clouds (6.5%) and cloud shadows (2.95%). Cloud shadow detection in the SSL model is relatively weak, with significant overlap with the cloud and background classes. In contrast, the

Copyright ©2024 by the International Astronautical Federation (IAF). All rights reserved.

SL model exhibits an improved performance, with background detection at 49.01%. Furthermore, cloud shadows are better recognized (6.72%) with reduced confusion. The better performance of the SL model in this task can be attributed to the direct supervision provided during training, which likely allows the model to learn more precise boundary definitions between clouds, shadows, and background, minimizing ambiguity.

Table 2. Clouds Segmentation Performance Comparison.

Configuration	mIoU
SSL	70.65
SL	82.00

3.2 Floods Segmentation

The dataset used for flood segmentation, as detailed in Subsection 2.1 (“Datasets”), initially included four distinct classes: “no event,” “floods,” “floods trace,” and “permanent water.” However, differentiating between flood water and permanent water using multispectral imagery proved to be extremely challenging, if not nearly impossible. Consequently, we opted to aggregate these two classes into a single “water” class to simplify the segmentation task and improve the model's performance. Globally, the class distribution of pixels after the merge of the two water classes is as follows:

- No event: 75.12%.
- Water: 21.45%.
- Floods Trace: 3.43%.

We used 14805 images for the training process of the application heads and 4643 images for testing.

Table 3 presents quantitative performance comparisons for the floods segmentation task using both the SSL and SL configurations.

Table 3. Floods Segmentation Performance Comparison.

Configuration	mIoU
SSL	91.83
SL	92.96

In Figure 4 (second column) are reported the confusion matrices computed for the floods segmentation task. In Figure 4 (second column) are reported the confusion matrices computed for the floods segmentation task.

In the flood segmentation task, the SSL model shows strong performance in detecting the "no event" class (74.69%) but struggles with water detection, achieving 20.72% and significant confusion with the "no event" class (0.64%). Flood trace detection is notably weak at 2.53%, indicating that the model struggles to distinguish flood traces from other classes. The SL model offers a slight improvement, maintaining high accuracy for the "no event" class (74.53%) and increasing water detection to 21.06%. Flood trace detection, however, remains low at 2.09%. The marginal improvement in the SL model's performance could be due to the availability of labelled data that helps the model differentiate subtle features of water and flood traces, although the intrinsic complexity of flood patterns still poses a challenge.

3.3 Fires Segmentation

For the fire segmentation task, we utilized the dataset in its original form, without any aggregation of classes or modifications. However, it is crucial to highlight the significant class imbalance present in this dataset:

- No event: 80.53%.
- Burnt area: 19.43%.
- Active flames: 0.04%.

This imbalance arises because it is extremely rare to capture remote-sensed images directly over areas with active flames. Additionally, due to their intrinsic nature, accurately recognizing and labelling active flames is challenging, leading to their minimal representation in Sentinel-2 imagery.

We used 14876 images for the training process and 4233 images for testing.

Table 4 presents quantitative performance comparisons for the cloud segmentation task using both the SSL and SL configurations. In Figure 4 (third column) are reported the confusion matrices computed for the fires segmentation task. As shown from the matrices, the SSL model performs well in classifying the "no event" class (76.2%) but shows confusion with the burnt area class (4.32%). Burnt area detection reaches 13.44%, though much of it is misclassified as "no event." The model is almost entirely ineffective at detecting active flames, with an accuracy of just 0.01%. The SL model slightly improves performance, with a "no event" detection of 76.89% and a burnt area classification accuracy rising to 17.53%. However, active flame detection remains extremely poor (0.01%). The difficulty that both models face in distinguishing burned areas and active flames certainly stems from the visual similarities between these features and the background, as well as the challenge of capturing small regions of dynamic flames, which are extremely rare in the dataset, as discussed at the beginning of this subsection; this scarcity of features related to active flames is amply visible by noting how only 0.04% of the pixels in the entire dataset contain active flames.

It is also plausible that during the SSL training of the backbone feature extractor, very few images, if any, contain active flames inside it, thus making it difficult to learn semantically representative features of the phenomenon that, even in the case of SL configuration is extremely difficult to detect correctly.

Copyright ©2024 by the International Astronautical Federation (IAF). All rights reserved.

Fig. 4. Confusion matrices of segmentation tasks reporting the absolute percentage value of true and predicted classes. Columns of two matrices from left to right indicates: clouds segmentation, floods segmentation and fires segmentation. Rows of matrices from top to bottom indicates: SSL configuration, SL configuration.

Table 4. Fires Segmentation Performance Comparison.

Configuration	mIoU
SSL	81.93
SL	89.91

4. Discussion

Looking at the results presented in the previous section we notice how, in general, the SL generally model performs better than the SSL model across all tasks, particularly in the detection of clouds and burnt areas. The primary reason for this improvement, how we might expect, is the presence of labelled data in the SL configuration, which enables the model to learn clearer class distinctions during the end-to-end training of the whole architecture. In contrast, the SSL model, lacking explicit supervision (except for the small finetune of the task-specific heads), struggles with overlapping or visually ambiguous features, leading to more confusion between classes. However, neither model performs optimally in detecting smaller or more complex classes, such as flood traces or active flames, which suggests that both configurations could benefit from more advanced techniques or, even better, by training on datasets that better represent these features.

However, it is worth noting that, even though the SSL configuration performs slightly worse than the SL in terms of segmentation accuracy, it successfully addresses our first objective: performing all of the three tasks with overall high accuracy while reducing the computational complexity. The SSL model allows for parallel inference, enabling the simultaneous resolution of all three tasks (clouds, floods, and fires segmentation), this parallel approach offers a significant advantage in terms of efficiency, as it minimizes the need for sequential processing.

In Table 5, we provide an analysis of the latencies for computing the three tasks using both configurations (quantized, INT8 precision) on a 7W low-power edge-device with an accelerator specifically tailored for DL models, highlighting how the SSL setup offers reduced latency. Additionally, we present a comprehensive comparison of the energy-latency-quality trade-off demonstrating that while the SL configuration slightly edges out in quality, the SSL model excels in energy efficiency and latency. The SSL configuration maintains a stable latency even when adding a finite number of task-specific heads, this is because the tasks are executed in parallel, allowing the system to handle multiple tasks simultaneously without a significant increase in processing time. On the other hand, in the SL configuration, latency increases linearly with the addition of task-specific heads. Each time a new head is added, it must be executed sequentially, leading to two main drawbacks: (1) increased latency due to the serial execution of tasks and (2) higher memory consumption on board, as each task requires its own model instance (backbone + head). This creates inefficiencies, particularly in systems with limited computational resources, where managing multiple models simultaneously can lead to performance bottlenecks. Therefore, while the SL model might offer slight improvements in accuracy, the SSL configuration proves more scalable and efficient, especially when dealing with multiple tasks in real-time applications.

Copyright ©2024 by the International Astronautical Federation (IAF). All rights reserved.

Table 5. Quality metric and latency comparisons in performing the three segmentation tasks.

Configuration	mIoU (avg on all tasks)	Latency
SSL	81.47	37.47ms
SL	88.29	114.41ms

6. Conclusions

We presented the design of an onboard Edge-AI system capable of performing inference of multiple tasks in parallel, demonstrating its potential through three segmentation tasks, two of which were tested on custom novel datasets. The combination of our self-supervised learning technique and a modular design enables the creation of an efficient system that can address complex tasks with low latency and minimal computational demands. The ability to run multiple tasks simultaneously highlights the advantage of the SSL configuration, particularly in resource-constrained environments where parallel processing is essential. However, further research is required to refine the self-supervised training strategy to ensure that the SSL model achieves performance levels comparable to the SL configuration, in terms of quality metrics such as the mIoU.

Acknowledgements

This study was partially funded and supervised by the Italian Space Agency in the framework of the Research Day “Giornate della Ricerca Spaziale” initiative, through ASI contract N. 2023-23-U.0, within the programme on “Analisi dati e immagini”.

References

[1] B. Zhang, Y. Wu, B. Zhao, J. Chanussot, D. Hong, J. Yao, and L. Gao, “Progress and challenges in intelligent remote sensing satellite systems,” IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing, vol. 15, pp. 1814–1822, 2022.

[2] M. Ziaja, P. Bosowski, M. Myller, G. Gajoch, M. Gumiela, J. Protich, K. Borda, D. Jayaraman, R. Dividino, and J. Nalepa, “Benchmarking deep learning for on-board space applications,” Remote Sensing, vol. 13, no. 19, p. 3981, 2021.

[3] G. Giuffrida, L. Fanucci, G. Meoni, M. Batič, L. Buckley, A. Dunne, C. van Dijk, M. Esposito, J. Hefele, N. Vercruyssen 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, 2021.

[4] Krizhevsky, Alex, Ilya Sutskever, and Geoffrey E. Hinton, "Imagenet classification with deep convolutional neural networks." Advances in neural information processing systems 25, 2012.

[5] Long, J., Shelhamer, E., & Darrell, T. Fully convolutional networks for semantic segmentation. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 3431-3440, 2015.

[6] R. Girshick, J. Donahue, T. Darrell and J. Malik, "Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation," 2014 IEEE Conference on Computer Vision and Pattern Recognition, 2014.

[7] T. Chen, S. Kornblith, M. Norouzi, and G. Hinton, “A simple framework for contrastive learning of visual representations,” International conference on machine learning. PMLR, 2020.

[8] J.-B. Grill, F. Strub, F. Altch´e, C. Tallec, P. H. Richemond, E. Buchatskaya, C. Doersch, B. A. Pires, Z. D. Guo, M. G. Azar, B. Piot, K. Kavukcuoglu, R. Munos, and M. Valko, “Bootstrap your own latent: A new approach to self-supervised learning,” Neural Information Processing Systems, 2020.

[9] F. Alistair and M. Czerkawski. "Major TOM: Expandable Datasets for Earth Observation." arXiv preprint arXiv:2402.12095 , 2024.

[10] Aybar, C., Ysuhuaylas, L., Loja, J. et al., “CloudSEN12, a global dataset for semantic understanding of cloud and cloud shadow in Sentinel-2.” Sci Data 9, 2022.

[11] I. Ashraful, et al. "Self-supervised learning with local contrastive loss for detection and semantic segmentation." Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision, 2023.

[12] Y. Tang, K. Han, J. Guo, C. Xu, C. Xu, and Y. Wang, “Ghostnetv2: enhance cheap operation with long-range attention,” Advances in Neural Information Processing Systems, vol. 35, pp. 9969–9982, 2022.

[CEMS] Copernicus Emergency Management Service, “Copernicus Emergency Management Service - Mapping”, 2023. https://emergency.copernicus.eu/mapping/