Artificial Intelligence Based On-Board Image Compression for the Φ-Sat-2 Mission

Giorgia Guerrisi, *Graduate Student Member, IEEE*, Fabio Del Frate, *Senior Member, IEEE*, and Giovanni Schiavon

***Abstract*—The growing amount of data collected by Earth Observation (EO) satellites requires new processing procedures able to manage huge quantity of information. Artificial intelligence (AI) and deep learning (DL) can provide advanced information also because of their ability to extract valuable information from complex data. Thanks to specific hardware platforms, these algorithms can be used also in space, opening the possibility for new procedures for intelligent data processing. The European Space Agency Φ-Sat-2 mission was designed with the purpose of demonstrating the benefits of using AI in space by running AI-based applications on-board a CubeSat. We present here the convolutional autoencoder-based algorithm developed for on-board lossy image compression of the Φ-Sat-2 mission and provide a first benchmark addressing a real space mission and a new image compression end-to-end architecture based on AI. Image compression is a crucial application that allows to save transmission bandwidth and storage. In fact, images acquired by the sensor can be compressed on-board and sent to the ground where they are reconstructed. DL algorithms have already been successfully applied for image compression however performance degradation may occur in the context of a representative on-board environment. Therefore, besides analyzing the results for the local hardware environment, this article investigates the performance variation for the on-board setting. An additional piece of innovation is the introduction of an applicative metric for the evaluation of the compression to assess the applicability of the reconstructed images for other tasks. Such metric completes those more traditional based on the original-reconstructed image similarity.**

***Index Terms*—Artificial intelligence (AI), convolutional neural networks (CNNs), CubeSat, image compression, on-board processing.**

I. Introduction

ADVANCED satellites on-board processing operations are becoming crucial. They represent a valuable solution to handle the progressively increasing amount of remote sensing data collected by sensors. For example, they can select among the acquired information only the pieces with higher interest, or they can compress data before the transmission to the ground [1]. In this context, artificial intelligence (AI) and in particular deep learning (DL) represent a valid instrument. DL algorithms, such as deep neural networks (NNs), can extract information from complex data and meaningfully improve the operations on the space platforms [2]. On-board AI is therefore a subject of growing importance, with new studies and investigations also encouraged thanks to CubeSat missions, which are offering new attractive opportunities in space technologies and scientific applications due to their limited cost and development time [3]. For example, the recent Φ-Sat-1 mission developed by the European Space Agency (ESA) demonstrated that the use of AI in space and on-board small satellites is possible and can lead to actual benefits. In the Φ-Sat-1 mission, an on-board cloud detection algorithm based on a convolutional neural network (CNN) allows to save data transmitted to the ground because only the acquisitions with less than 70% of cloudiness are considered [4], [5]. Moreover, studies in this field are supported by hardware advances, such as field programmable gate arrays (FPGAs) accelerator, specifically designed for AI-based space applications [6], [7]. Advanced on-board processing based on AI has been explored so far for several applications as volcanic eruption detection in multispectral images [8] or oil spill identification from SAR images [9]. In the same context, image compression represents an interesting application as well. If the acquisitions are compressed on-board and then decoded on the ground, bandwidth for data transmission is saved. In addition, acquisitions can be stored as compressed data and decompressed only when needed.

More in general, image compression is the technique used to minimize the size of an image and two main approaches to image compression exist: lossy and lossless. In lossy compression part of the information stored in the original image is lost during the compression process, however the size of the resulting image is significantly reduced compared to lossless compression techniques. In fact, lossless compression does not considerably reduce the file size, but keeps the image quality as higher as possible. Among lossless image compression procedures, arithmetic coding and Huffman coding are the most used, while for lossy compression joint photographic experts group (JPEG), JPEG 2000, and vector quantization are the most common techniques [10]. In addition to traditional image compression algorithms, DL algorithms have also been successfully applied in this field and are under continuous development. Particularly, CNNs were applied for lossy image compression purposes by several authors, who mainly used encoding–decoding structures for image compression and reconstruction [11], [12]. The most used NN types for image compression are autoencoders (AEs),

Manuscript received 30 March 2023; revised 15 May 2023 and 22 June 2023; accepted 5 July 2023. Date of publication 18 July 2023; date of current version 12 September 2023. *(Corresponding author: Giorgia Guerrisi.)*

The authors are with the Department of Civil Engineering and Computer Science Engineering, "Tor Vergata" University of Rome, 00133 Rome, Italy (e-mail: giorgia.guerrisi@uniroma2.it; fabio.del.frate@uniroma2.it; schiavon@uniroma2.it).

Digital Object Identifier 10.1109/JSTARS.2023.3296485

variational autoencoders (VAEs) [13], [14], generative adversarial networks [15], and recurrent neural networks [16].

Despite the growing interest in CNN-based techniques for image compression, still little has been written about the compression of remotely sensed images, especially with DL algorithms to be run on-board the acquisition platforms. Related to this topic, Goodwill et al. [17] proposed an algorithm that combines CNN and JPEG compression, and the CNN is based on the one described in [14]. Li and Liu [18] proposed a method that uses CNN and nonnegative Tucker decomposition (NTD), that is a tensor decomposition, to perform multispectral image compression. CNN is used in combination with a 3-D-discrete cosine transform to reduce the multispectral image to a small-scale version in order to apply the tensor decomposition on-board. A complexity-reduced VAE to be run on-board satellites was developed by de Oliveira et al. [19] and utilized by the same authors also in [20] for the compression-denoising task. The VAE is based on the model proposed in [13] but the authors performed a reduction of the number of filters composing the convolutional layers and a simplification of the entropy model in order to reduce the complexity of the original architecture. It must be noted that except for [17], no reference to specific hardware for on-board processing is made in the other articles.

In this article, we present an AI-based image compression algorithm developed to be run on-board a CubeSat, and we analyzed the performance in a simulated on-board context. The compression algorithm is based on a Convolutional AutoEncoder (CAE), i.e., an AE with convolutional layers. This algorithm has been developed in the framework of the Φ-Sat-2 mission by ESA. The mission aims at demonstrating the capability and advantages of running AI in space. A 6U CubeSat platform will carry on-board a multispectral optical camera and the Intel Movidius$^{\text{TM}}$ Myriad$^{\text{TM}}$ 2 AI processor. The on-board computer will be capable to run different AI-based applications that can be developed, installed, validated, and operated on the spacecraft during flight using a simple user interface [21], [22], [23], [24]. On-board processing involves the migration of procedures and algorithms into environments characterized by reduced computational capabilities, with a hardware which may be completely different from that used for the determination of the AI model. In this context, the evaluation of the possible decrease of the performance after the on-board implementation can be an interesting aspect to be investigated. To our knowledge, so far, no paper deal with this issue in scientific literature especially as far as image compression by means of AI algorithms is concerned. For this reason, an original purpose of this article is to compute the cost, in terms of performance, of the migration of the AI module from the ground to the on-board simulated operational environment. Processing time and accuracy in the image reconstruction are the main variables examined in the analysis. In particular, and this is another innovative aspect of the article, the quality of the reconstructed image is evaluated not only using standard metrics, i.e., peak signal-to-noise ratio (PSNR) and structural similarity index measure (SSIM) [25], [26], but also with an applicative metric that aims at evaluating

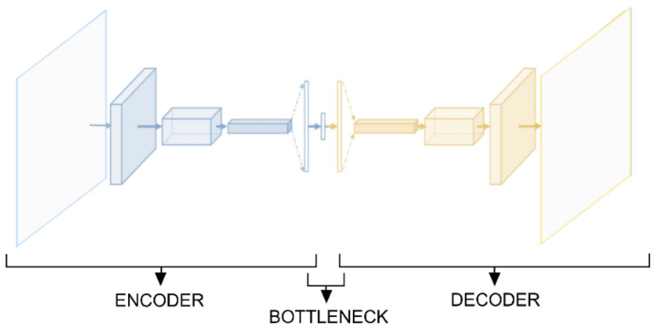

Fig. 1. General structure of a CAE, consisting of encoder, bottleneck, and decoder.

the practical use of the reconstructed images. The images compressed and reconstructed through the CAE were indeed used in a real application scenario, that is semantic segmentation.

The rest of the article is organized as follows. Section II describes the methodology, illustrating in detail the CAE architecture, the dataset, the preprocessing operations, and the metrics used for the evaluation of the results. Section III reports the results achieved in the considered hardware environments and provide a comparison among them. Section IV reports the discussion. Finally, Section V concludes this article.

II. Methodology

A. CAE Model

The model architecture is based on AEs, an NN topology with a symmetrical structure commonly used for feature learning and data compression [27], [28], [29]. AEs consist of an encoding part and a decoding part, with one or more hidden layers, and an internal bottleneck layer with fewer parameters than either input or output that forces a compressed knowledge representation of the original input [30]. Essentially, AEs transform input data to a lower dimensional representation and ensure the reconstruction of the input based on these characteristics. Due to the inner property of AEs, the output reconstruction is not equal to the input because there are always data losses during reconstruction. The simplest AE configuration is the multilayer perceptron AE, that is a feedforward NN. CAEs are AEs composed of convolutional and deconvolutional layers instead of fully connected layers. Compared with standard AEs, CAEs can lead to even better results because convolutional layers can capture spatial correlations in the images and have demonstrated a better ability in extracting features from images [31], [32], [33]. This model is summarized in Fig. 1, where the encoder is shown in blue and the decoder in yellow. The figure shows the encoder with the convolutional layers that are flattened to return dense layers. In the middle, there is the bottleneck layer. In the decoder, deconvolutional layers return back the output with the same dimension as the input.

The encoder extracts the compressed representation of the original input, while the decoder returns it back. Between them, the bottleneck layer determines the compression ratio (CR) [33].

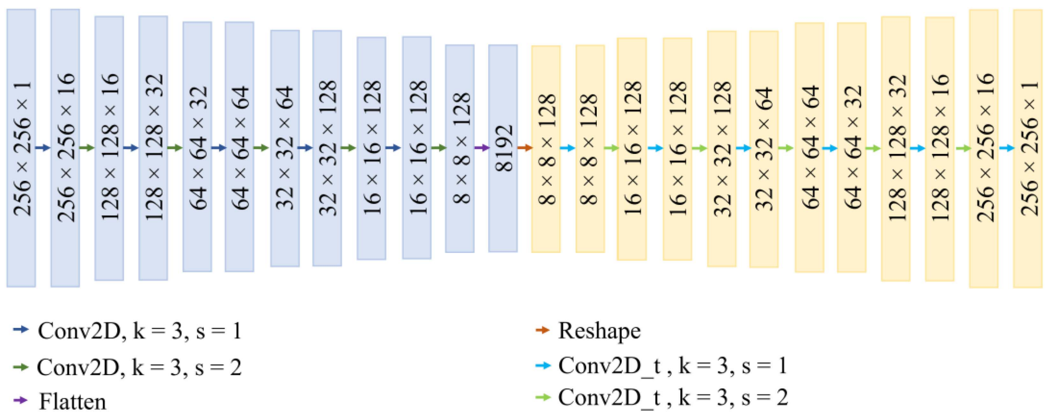

Fig. 2. CAE structure. Encoder is shown in blue and decoder is shown in yellow. Each block reports the output shape as height × width × depth. The arrows are associated with the operation reported at the bottom. Conv2D indicates convolutional layer, *k* the kernel size, *s* the stride, and Conv2D_t transposed convolutional layers.

CR is determined by the dimension of the bottleneck layer, hence by the architecture of the model. CR is defined as (1) where input size is in this case the size of the input image in terms of pixels and bottleneck size corresponds to the number of units in the bottleneck layer

$$\mathrm{CR} = \frac{\text{Input size}}{\text{Bottleneck size}}. \tag{1}$$

A more detailed representation of the proposed CAE model is shown in Fig. 2. The final definition of the model was determined taking into account the requirements imposed by the mission, especially in terms of memory footprint and RAM usage. The encoder (in blue in Fig. 2) includes blocks of convolutional layers, and the innermost bottleneck layer, that determines the size of the compression. Symmetrically, the decoder part (in yellow in Fig. 2) includes transposed convolutional blocks. In the encoder, convolutional layers (indicated as Conv2D in Fig. 2) progressively extract deeper feature maps through an increasing number of filters. Then, the output of the last convolutional layer is flattened to return the compressed representation of the image. The encoder is composed of 10 convolutional layers followed by leaky ReLU activation function. Each layer has a kernel size of 3×3 (*k* in Fig. 2) with an increasing number of filters (16, 32, 64, and 128) and an alternating stride (*s* in Fig. 2) of 1 and 2, the latter for the downsampling convolutional layer. The last layer of the encoder is a flatten layer which returns 8192 units. Compared to the original input dimension, that is $256 \times 256 \times 1$, the bottleneck returns a CR of 8. In the decoder, the reverse process is performed, returning the reconstructed image as final output. The decoder reshapes the flatten layer into a 3-D tensor of size $8 \times 8 \times 128$. Then, transposed convolutional blocks (Conv2D_t in Fig. 2) with leaky ReLU activation function return an output of size $256 \times 256 \times 16$. The last block gives back the output with size $256 \times 256 \times 1$ and has a linear activation function.

After the training phase, the encoder and decoder parts are used separately. The encoder part of the network is executed on-board: it takes the acquired image as input and returns the compressed representation obtained by means of the bottleneck layer. The reduced vector encodes the original image and can be stored and sent to the ground. Here, the decoder part returns the reconstructed image.

B. Dataset

The dataset used for training and validating the algorithm is based on Sentinel-2 acquisitions, which were converted to simulate the final Φ-Sat-2 acquisitions through a processing chain that will be described soon in this paragraph. The choice to use Sentinel-2 data is due to several reasons: first, the large availability of acquisitions and the free and open access policy. Then, the similarity between data acquired by the multispectral instrument (MSI) on-board the Sentinel-2 satellite and the Φ-Sat-2 multispectral camera. The input dataset consists of 10 Sentinel-2 acquisitions at L1C processing level, reported in Table I. The table also specifies the acquisitions used for the training of the model and those used for the validation phase, i.e., the inference phase. Six of them are used to compose the training dataset and the remaining four to compose the validation dataset. The acquisitions are captured over Europe, America, and Asia, between 2019 and 2020. Sentinel-2 L1C products report top of atmosphere reflectance measurements stored as 16-bit integers. The radiometric resolution of Sentinel-2 MSI is 12 bits, but MSI measurements are converted to reflectance and stored as 16-bit integers.

The selected images have the following characteristics: they show urban and suburban areas, they are mainly free from cloud and haze and exclude large sea areas. The choice of using images with the presented characteristics aims to optimize the performance of the model in the case of complex scenarios. However, scenes with a percentage of cloud and marine cover are also considered, especially in the validation phase, in order to assess the model performance in these particular cases. A couple of examples extracted from the training set are shown in Fig. 3, while Fig. 4 reports two examples extracted from the validation set. As shown in Fig. 4, images for the validation dataset contain cloudy areas, large sea areas, large urban areas, vegetated land, and suburban areas.

TABLE I
SENTINEL-2 ACQUISITION IDENTIFIERS USED TO BUILD THE DATASET

File ID	Usage
S2A_MSIL1C_20191226T103431_N0208_R108_T32VNH_20191226T110451	Training
S2A_MSIL1C_20200108T104421_N0208_R008_T31TFJ_20200108T111207	Training
S2A_MSIL1C_20200109T152631_N0208_R025_T18NWL_20200109T185317	Training
S2A_MSIL1C_20200207T104211_N0209_R008_T31UGS_20200207T112740	Training
S2A_MSIL1C_20200305T171151_N0209_R112_T14SPB_20200305T204641	Training
S2A_MSIL1C_20200207T104211_N0209_R008_T31UGT_20200207T112740	Training
S2A_MSIL1C_20191219T104441_N0208_R008_T32UMG_20191219T111510	Validation
S2A_MSIL1C_20191211T030121_N0208_R032_T49QGE_20191211T055310	Validation
S2A_MSIL1C_20200106T100401_N0208_R122_T33TTG_20200106T103423	Validation
S2A_MSIL1C_20191211T030121_N0208_R032_T49QGG_20191211T055310	Validation

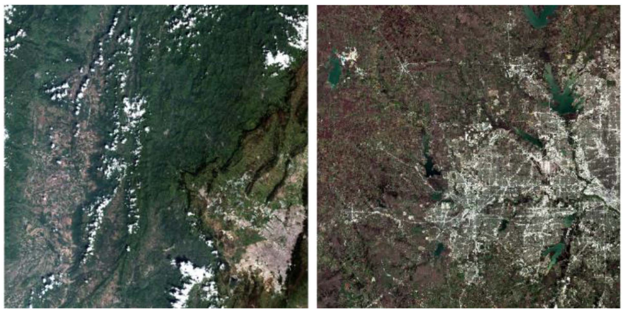
Fig. 3. Sentinel-2 acquisitions extracted from the training set in RGB composition.

Fig. 4. Sentinel-2 acquisitions extracted from the validation set in RGB composition.

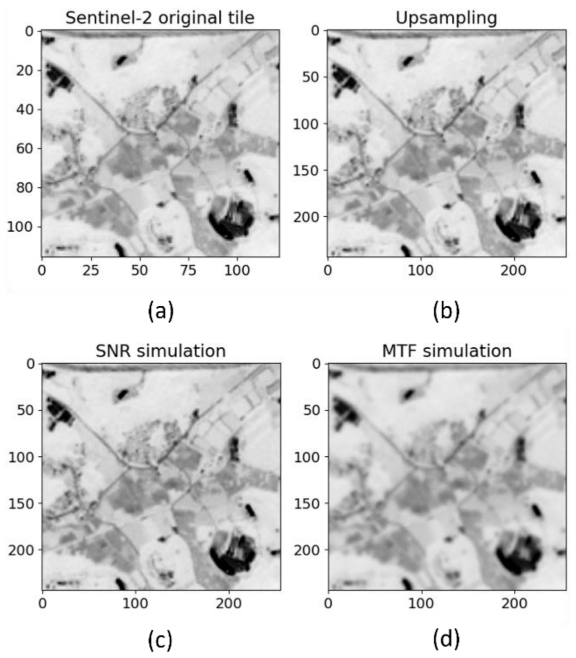

Fig. 5. Sentinel-2- Φ-Sat-2 dataset conversion. (a) Sentinel-2 original tile. (b) Output after upsampling. (c) Output after SNR simulation. (d) Output after MTF simulation.

The images were preprocessed to simulate the characteristics of the Φ-Sat-2 imager. The steps to be performed for the conversion include the selection of the matching spectral bands and the simulation of the spatial resolution, signal-to-noise ratio (SNR), and modulation transfer function of the Φ-Sat-2 camera. Fig. 5(a) shows a patch with a dimension of 128 $\times$ 128 pixels extracted from a Sentinel-2 acquisition in band 4 and the corresponding Φ-Sat-2 simulated one *(d)*. The output after upsampling and SNR simulation are shown in Fig. 5(b) and (c), respectively.

C. Training Dataset Preprocessing

Before the training of the network, the training dataset was preprocessed and the preprocessing steps can be summarized as: 1) band selection, 2) tiling, 3) saving, and 4) data type conversion and normalization. The preprocessing flow accepts as input the Φ-Sat-2 simulated single band acquisition and returns as output a tensor representing the input to the encoder. Related to band selection, only bands number 2, 3, 4, and 8 were selected, i.e., the Sentinel-2 bands with the highest spatial resolution of 10 m. In this way, the model will be trained with images that are closer in terms of ground sample distance (GSD) to the final acquisitions, and in general with more complex data, with the expectation of obtaining good results even when the Φ-Sat-2 images will be used. At the tiling stage, the single-band image was divided into subimages, i.e., patches, with a dimension of 256 $\times$ 256 pixels so that the input of the encoder has a size of *patch height* $\times$ *patch width* $\times$ *image depth*. The latter is equal to 1 in this case, since single band images were considered. During the experimental phase, different patch sizes were considered, but the size of 256 $\times$ 256 was chosen as it represents the best compromise between power consumption and processing time, and accuracy. Destripy operations could be executed on the ground as postprocessing operations to face eventually stripy phenomena. Removing stripes on-board indeed would imply an increase in processing time and power consumption. Moreover, the results in terms of standard and applicative metrics confirm that the reconstructed images can be used with good results. Then, the patches extracted from the single-band image were saved and, before the training, the pixels values were converted to float32 data type and each patch was normalized.

After preprocessing, the training dataset was randomly split into three distinct parts, training, validation, and test sets, with a proportion of 70%, 20%, and 10%, respectively. The training and validation datasets were used during the training phase, while the test dataset to evaluate the performance of the trained model.

D. Validation Dataset Preprocessing

It should be noted that the validation dataset here described is the one used during the inference phase. The preprocessing flow accepts as input the single-band acquisition and returns as output a tensor representing the input to the encoder. In contrast to the preprocessing of the training dataset, all the bands acquired by the imager were considered for the validation dataset. Moreover, the normalization step does not include the data type conversion since the pixels values of the validation images were maintained as 16-bit floating point data. So the preprocessing steps can be summarized as: 1) tiling, 2) saving, and 3) normalization.

E. Evaluation Metrics

The implemented compression procedure is a lossy procedure. This means that part of the information stored in the original image is lost during the compression process. Image quality evaluation methods are used to evaluate this loss by calculating the similarity between the original image and the compressed one. PSNR, that depends on mean squared error (MSE), and SSIM are among the most popular index for image quality evaluation, as highlighted also in the work reported in the introduction, and they were used to evaluate the performance of our CAE model.

For a reference image f and a test image g, having dimension $M \times N$, represented with a dynamic range of pixels intensity L, PSNR is defined as

$$\mathrm{PSNR}\,(f,\ g) = 10\log_{10}\left(\frac{L^2}{\mathrm{MSE}\,(f,g)}\right). \tag{2}$$

Equation (3) defines the MSE, where i, j is the generic pixel at row i and column j

$$\mathrm{MSE}\,(f,g) = \frac{1}{MN}\sum_{i=1}^{M}\sum_{j=1}^{N}(f_{ij} - g_{ij})^2. \tag{3}$$

If the reconstructed image is exactly same as original, then MSE approaches 0 and PSNR approaches infinity. This means that higher PSNR values indicate higher image quality [26]. L in (2) is equal to 1 in this case because input and output images are represented as floating-point data type normalized in the 0–1 interval, as described in Section II-C.

SSIM is related to the perception of image quality by the human eye [34]. This index models image distortion as the combination of three distortion factors, which are luminance l, contrast c, and structure s. For a reference image f and a test image g, SSIM is defined as (4), where l, c, and s, respectively, depend on the mean luminance, standard deviation, and covariance between the two images. $\alpha > 0$, $\beta > 0$, and $\gamma > 0$ control the relative significance of each of the three terms of the index; in this implementation, $\alpha = \beta = \gamma = 1$

$$\mathrm{SSIM}\ \ (f,g) = [l\,(f,g)]^{\alpha}\ \cdot [c\,(f,g)]^{\beta} \cdot [s\,(f,g)]^{\gamma}. \tag{4}$$

SSIM varies between 0 and 1: 0 means no correlation between the images f and g, while 1 represents a perfect match for f and g [35]. A close relationship exists between MSE and SSIM, as proved in [26] and [35].

SSIM was preferred for the estimation of the reconstruction quality, since it is characterized by a fixed value range. The PSNR, with $L = 1$, could also be considered however it does not have a fixed maximum value and the optimum intervals found in the literature refer to integer values images with a bit depth of 8, 12, or 16 [36], [37].

In addition, the algorithm performance was evaluated with regard to the applicability of the reconstructed images to common application cases. This kind of evaluation can provide an actual measure of the performance of the algorithm, not only in quantitative values. A semantic segmentation algorithm for the detection of building was implemented, using both the original and reconstructed images as test dataset, as described in detail in Section III-E.

III. Results

This section describes the obtained results. Section III-A provides details on the training process related to the dataset and training environment, as well as the results achieved after the training is complete. Then, Sections III-B and III-D describe the results evaluated during the inference phase for the validation dataset for three different environments. They are characterized by three different hardware, which are the graphic processing unit (GPU) NVIDIA GeForce GTX 1650, the vision processing unit (VPU) Intel Movidius$^{\mathrm{TM}}$ Myriad$^{\mathrm{TM}}$ 2, and the central processing unit (CPU) Intel Core$^{\mathrm{TM}}$ i7-6700. Special attention is paid to the evaluation of the processing time since the latter was particularly affected by the migration of the model from the offline to the on-board environment. Finally, Section III-E illustrates the results obtained for the real-application scenario, i.e., the semantic segmentation application.

A. Model Training

The training dataset consists of 48 640 patches with a dimension of 256 pixels $\times$ 256 pixels $\times$ 1. Among them, 34 048 are for training, 9728 are for validation, and 4864 are for testing. The input size in terms of pixels is equal to 65 536 and the bottleneck size is 8192, resulting in a CR equal to 8. The output of the compression is represented by an array stored as float16 data type (the same as the input data). The vector has 8192 elements, as determined by the bottleneck layer.

The model was developed and trained in Python 3.6 with TensorFlow 2.1.0 and the training was stopped using the early stopping algorithm. Training configuration is as follows: the optimizer is Adam, the learning rate starts from 10^{-4} and decreases

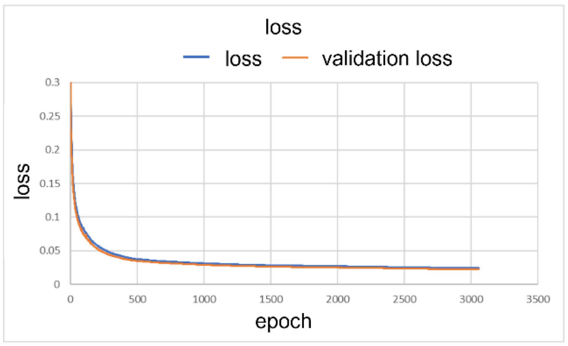

Fig. 6. Training and validation loss.

TABLE II
TRAINING PERFORMANCE

	Loss	SSIM	MSE
Training set	0.0244	0.9756	2e–4
Validation set	0.0225	0.9775	2e–4
Test set	0.0234	0.9766	2e–4

if no improvement in validation loss is obtained for a given number of epochs, until it reaches a value of 10^{-5}. We used SSIM as loss function. SSIM has already been successfully used as training loss [38] and in this case led to better results than other common loss function such as MSE.

The SSIM loss function is defined as

$$\text{SSIMloss}\left(y_{\text{true}},\ y_{\text{pred}}\right) = 1 - \frac{1}{M}\sum_{i=1}^{M} \text{SSIM}\left(y_{\text{true } i},\ y_{\text{pred } i}\right). \tag{5}$$

As defined in the standard SSIM implementation [34], the metric is computed locally within a local window defined as a Gaussian filter with size 11 × 11. Thus, in (5), y_{true}, y_{pred} are the input and the reconstructed images, $y_{\text{true } i}$, $y_{\text{label } i}$ are the image contents at the ith local window, and M is the number of local windows in the image. Finally, the similarity index is transformed into dissimilarity index by adding one minus the similarity index.

Fig. 6 shows the trend of the loss and validation loss during training. The PSNR and MSE are defined as in (2) and (3), respectively. The SSIM metric is defined as

$$\text{SSIMloss}\left(y_{\text{true}},\ y_{\text{pred}}\right) = \frac{1}{M}\sum_{i=1}^{M} \text{SSIM}\left(y_{\text{true } i},\ y_{\text{pred } i}\right). \tag{6}$$

Fig. 7 shows the trend of the SSIM and MSE metrics. Table II reports the values of the loss function and the SSIM and MSE metrics at the end of the training process for the training, validation, and test sets. The loss values confirm, as also shown in Fig. 6, the good fit achieved during the training process. The SSIM is around 0.97 for all the subsets of the dataset, reaching a value of 0.9766 for the test set. MSE is, on average, about 2e−4 for the training, validation, and test sets.

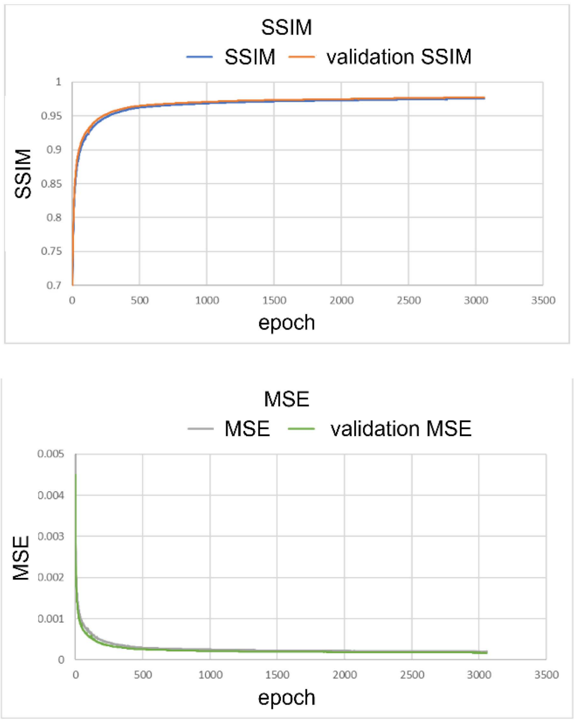

Fig. 7. Training and validation SSIM and MSE.

TABLE III
MODEL METRICS FOR TEST DATASET

Metrics	Value
CR	8
SSIM	0.977
PSNR	38.58
MSE	2e–4

Fig. 8 shows four examples extracted from the test set. The original 256 × 256 tile is shown on the left, while the reconstructed tile is shown on the right. For each tile, the values of the metrics are also reported. Table III reports the values for SSIM, PSNR, and MSE computed as an average over all patches of the test set.

B. Results—GPU Environment

The GPU environment comprises the NVIDIA GeForce GTX 1650 with 4 GB RAM. Fig. 9 shows the histogram of the SSIM calculated for each patch in each image of the validation dataset. A zoomed portion of the histogram is also shown, in order to highlight the distribution of the SSIM values with frequencies between 0 and 100. As described in Section II-D, only two tiles for each Sentinel-2 size acquisition are considered, resulting in a total of 2048 patches per band (14 336 patches for all bands).

The green line indicates the average SSIM computed as average value over all the patches belonging to the validation set that is 0.9651. Compared to the value achieved during the training process, that is 0.9766, the average SSIM decreases

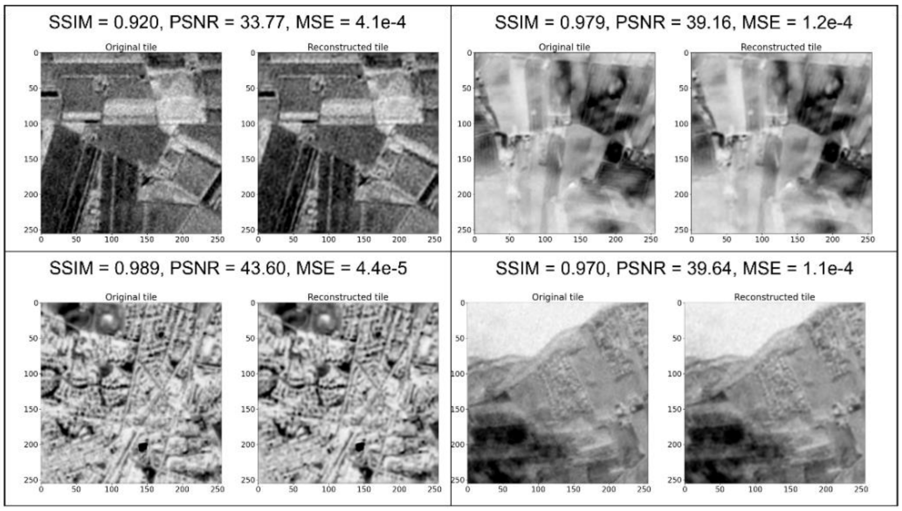
SSIM = 0.920, PSNR = 33.77, MSE = 4.1e-4
SSIM = 0.979, PSNR = 39.16, MSE = 1.2e-4
SSIM = 0.989, PSNR = 43.60, MSE = 4.4e-5
SSIM = 0.970, PSNR = 39.64, MSE = 1.1e-4
Original tile
Reconstructed tile

Fig. 8. Original and reconstructed tiles extracted from the test set.

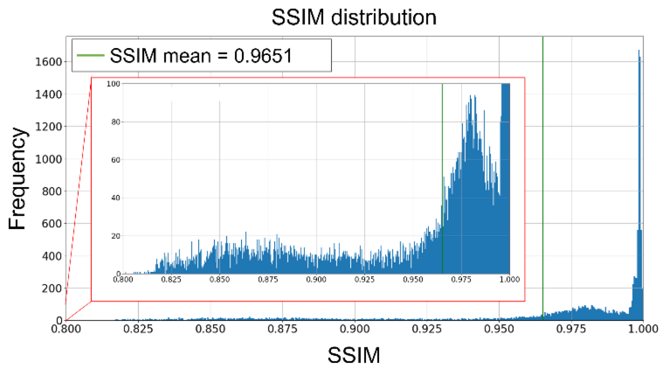

Fig. 9. SSIM distribution for the validation set for the GPU environment. The red box shows a detailed view for frequency values between 0 and 100.

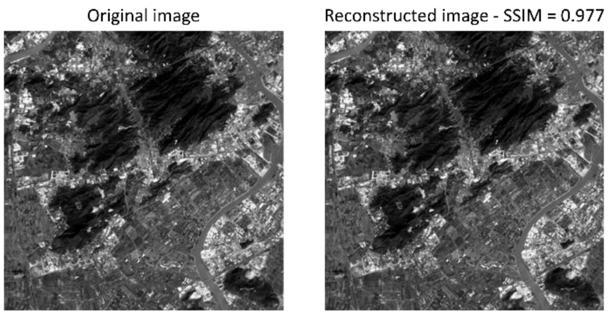

Fig. 10. Validation image T49QGE_20191211T030121_B03_tile_4097-8194.

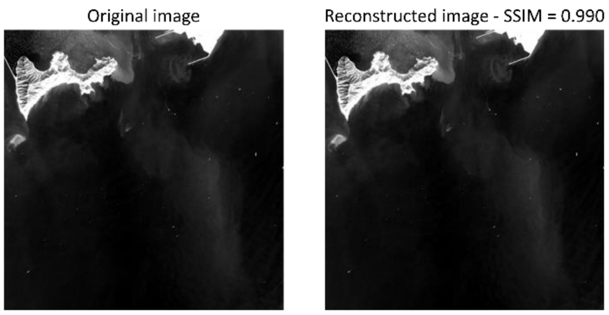

Fig. 11. Validation image T49QGE_20191211T030121_B05_tile_4097-16388.

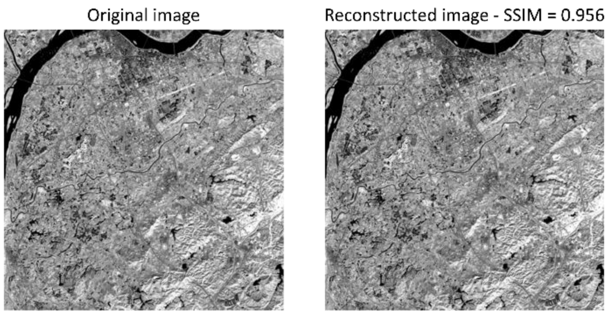

Fig. 12. Validation image T49QGG_20191211T030121_B08_tile_0-16388.

slightly by about 0.01. As shown in the histogram, the absolute minimum value is around 0.8, while the maximum is.

Figs. 10–12 show the original and reconstructed validation images in one single band, with the related SSIM obtained for the entire image. The CR is equal to 8, the original size is 32.00 MB, and the compressed size is 4.00 MB. In addition, the processing time is also evaluated. The processing time is the time required by the application to compress a single band of the acquisition. The time required by the model to compress a single band of the acquisition is, on average, 0.95 s.

As a comparison, the results obtained in terms of SSIM by means of the CAE algorithm are compared with those obtained with JPEG 2000 compression algorithm. Indeed, JPEG 2000 is the latest compression standard emerged from the JPEG, performs in general better than JPEG, and is also used to compress independently the bands of Sentinel-2 multispectral images [39].

TABLE IV
JPEG–CAE PERFORMANCE COMPARISON

	SSIM — JPEG 2000 compression	SSIM — CAE compression
Fig. 10	0.944	0.977
Fig. 11	0.983	0.990
Fig. 12	0.947	0.956

The same input image was compressed with the encoder part of the CAE and with the JPEG 2000 algorithm using a quality factor that returns an image eight times smaller than the original. The results are reported in Table IV for the three validation images reported in Figs. 10–12.

C. Results—VPU Environment

VPUs are a class of hardware specifically designed to increase the speed of visual processing, such as CNNs, ensuring a balance between power efficiency and compute performance. The Intel Myriad 2 is a commercial off-the-shelf VPU and represents the best accelerator in this category. It is characterized by high processing efficiency, high streaming throughput, small size, and low-cost. Moreover, the Myriad 2 processor has already passed the radiation tests at CERN, making it suited for space applications [5], [40], [41], [42]. For these reasons, the Myriad 2 AI processor has been adopted in the ESA's Φ-Sat-1 and Φ-Sat-2 missions [4], [22].

As explained in the introduction, the encoder was developed to run on board the nanosatellite where the processing hardware is the Myriad VPU. Therefore, the results obtained by running the encoder and decoder on the GPU were compared with those obtained by running the encoder on the Intel Myriad 2 connected via USB to the local machine. The model must be preliminary converted using the OpenVINO toolkit [43]. This toolkit is specifically designed to optimize and deploy DL models for the supported hardware platforms, including the Myriad 2 VPU. In short, the toolkit converts the trained network into an intermediate representation readable by the inference engine [44]. Then, the inference phase is performed on the Myriad 2. The time required to compress a single-band image is 10.92 s, which is considerably higher than the time required to compress the same image on the GPU. Related to the SSIM, the results obtained are almost the same.

Similar to Figs. 9 and 13 shows the histogram of the SSIM calculated for each patch in each image of the validation dataset with the relative enlargement for frequencies between 0 and 100. The green line indicates the average SSIM that is 0.9647, which differs by 0.0004 compared to the average SSIM value calculated on the GPU.

Figs. 14 and 15 show two examples of test images with the respective reconstructions obtained by means of the GPU and the VPU. The whole image is shown in the first row, while the second row shows the enlargement of the area bounded by the red box. Figs. 14 and 15(a) show the original image, while the reconstructed images obtained after compression on the GPU and VPU are shown in figures (b) and (c), respectively. The SSIM

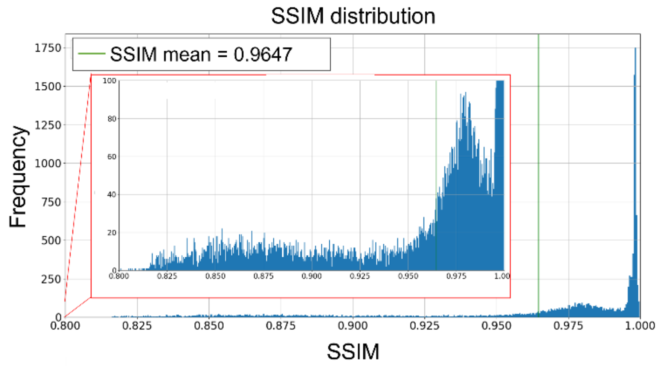

Fig. 13. SSIM distribution for the validation set for the VPU environment. The red box shows a detailed view for frequency values between 0 and 100.

TABLE V
PERFORMANCE COMPARISON

	GPU	VPU	CPU
Power consumption	83 W	1.4 W	45.5 W
Processing time	0.95 s	10.92 s	1.9 s
Energy efficiency	0.011	0.058	0.010

value is also reported for the two test images and is almost the same for the two scenarios.

D. Results—CPU Environment

In order to compare the results obtained for the VPU environment, the CPU environment was also considered. Indeed, the model optimized through the OpenVino framework can be executed on different Intel devices, so it was run on the Intel Core$^{\mathrm{TM}}$ i7-6700 Processor. This makes it possible to compare the performance using the same dataset and the same model converted following the same processing operations and data flow. Table V summarizes the performance in terms of power consumption, expressed in Watt (W), energy efficiency, expressed as GFLOPS per Watt, and processing time, expressed in seconds (s), for the three different environments. As expected, the VPU power consumption is considerably lower than GPU and CPU, with a value of 1.4 W. On the other hand, the highest power consumption is related to the GPU environment, with a value of 83 W. The energy efficiency was computed as the ratio between the floating point operations per second and the power consumption, and the VPU resulted as the most efficient hardware. The processing time is the time required by the encoder to compress a single acquisition with the simulated Φ-Sat-2 size. It almost doubles switching from the GPU to the CPU environment, and considerably rises from the CPU to the VPU environment. For the interpretation of this result, we put forward the following explanations. The main reason could be related to the time needed to transfer the image to the VPU via the USB port. This time is not consumed by the CPU instead. Second, the CPU and VPU hardware are completely different, and this

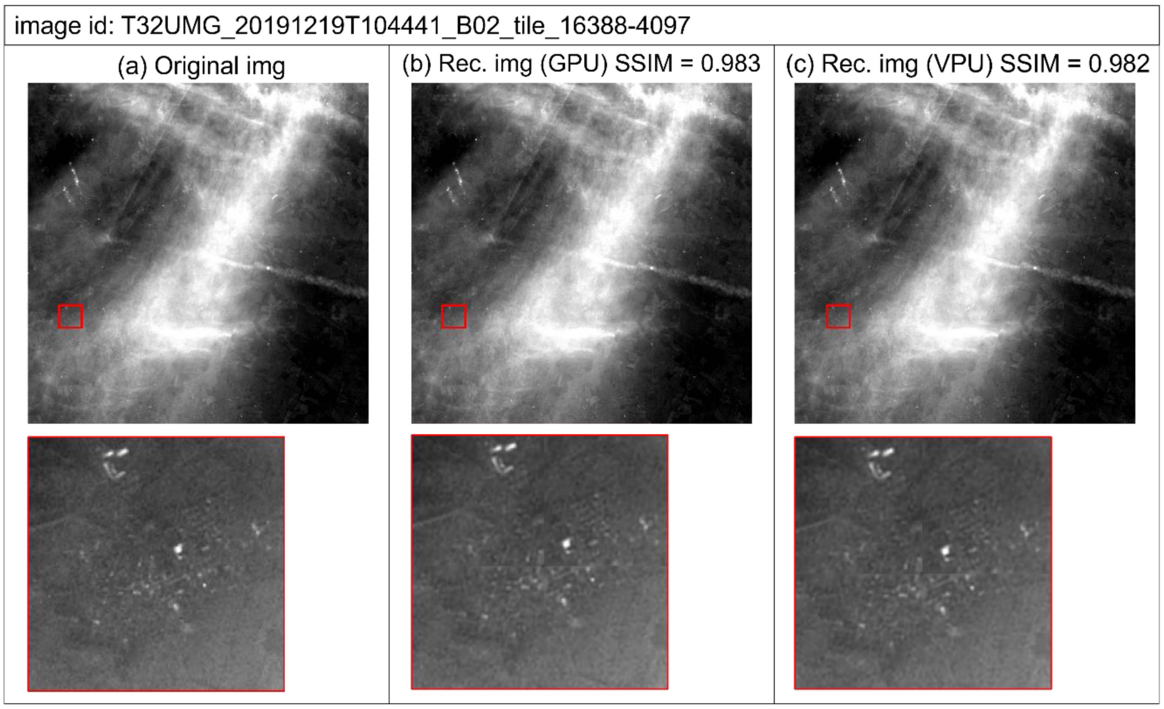
image id: T32UMG_20191219T104441_B02_tile_16388-4097
(a) Original img
(b) Rec. img (GPU) SSIM = 0.983
(c) Rec. img (VPU) SSIM = 0.982

Fig. 14. (a) Original test image; (b) reconstructed image obtained by encoding on GPU; and (c) reconstructed image obtained by encoding on VPU. The red boxes indicate the location of the zoomed portion shown below.

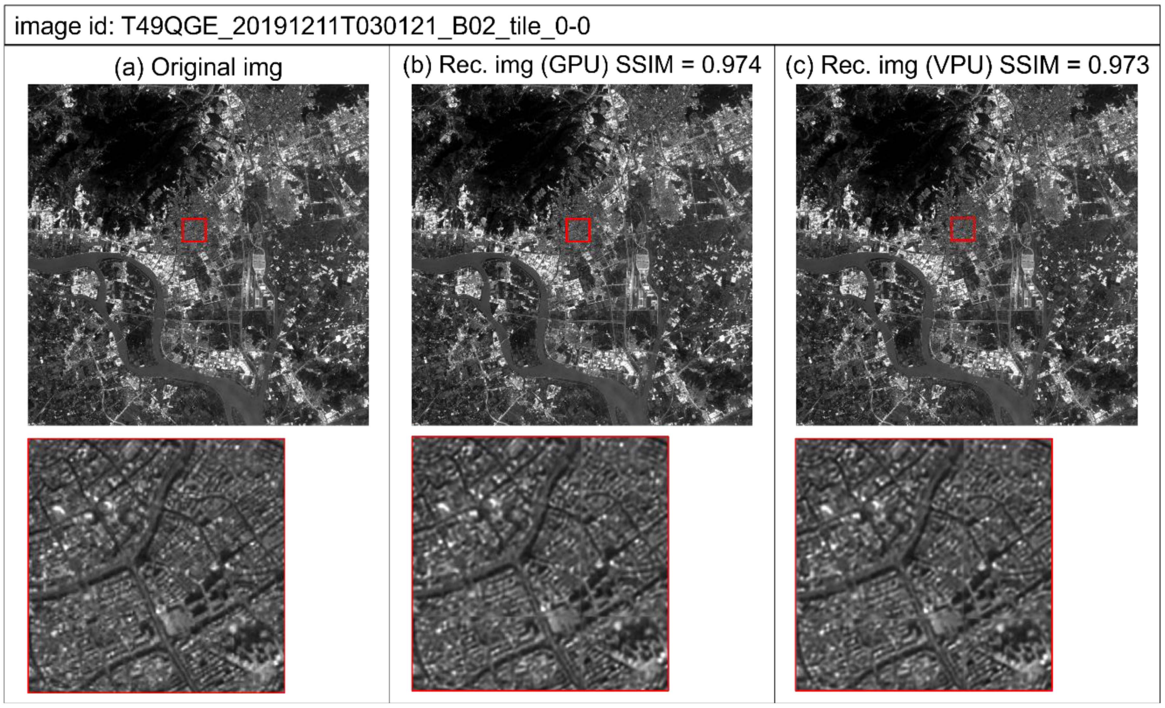
image id: T49QGE_20191211T030121_B02_tile_0-0
(a) Original img
(b) Rec. img (GPU) SSIM = 0.974
(c) Rec. img (VPU) SSIM = 0.973

Fig. 15. (a) Original test image; (b) reconstructed image obtained by encoding on GPU; and (c) reconstructed image obtained by encoding on VPU. The red boxes indicate the location of the zoomed portion shown below.

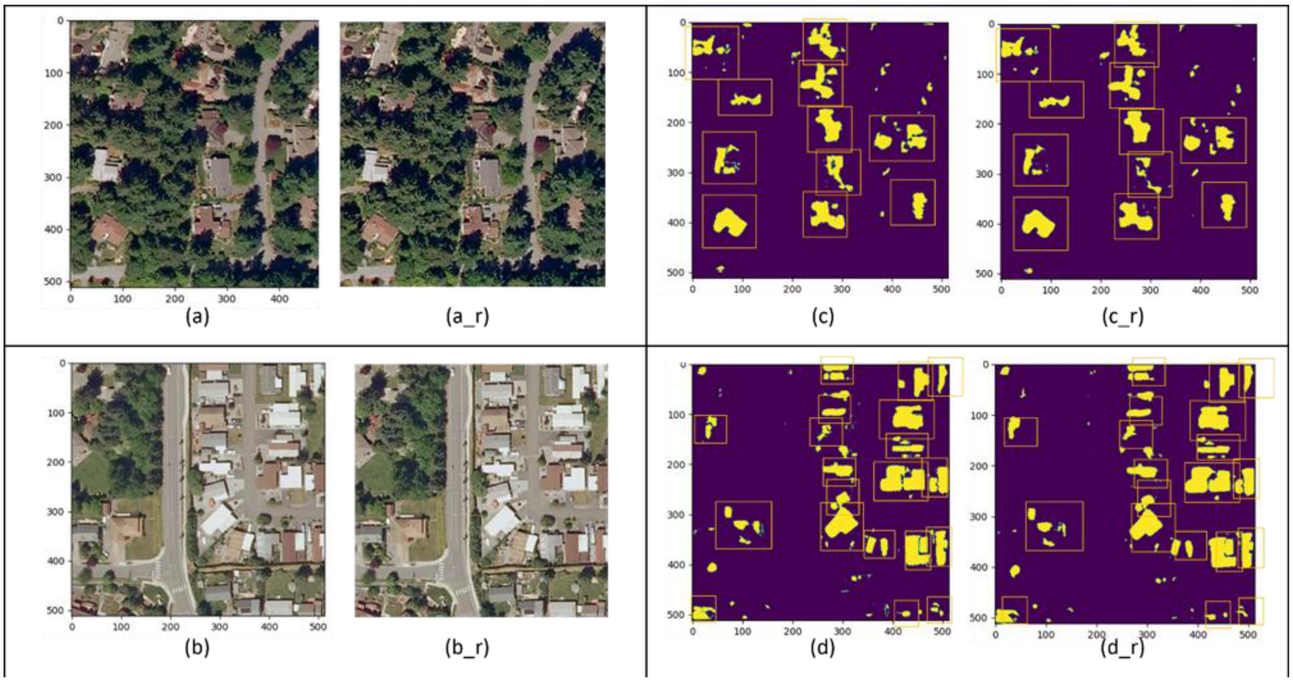

Fig. 16. Results for the semantic segmentation application. (a), (b) Original tiles extracted from two test images of the Inria aerial image labeling dataset. (a_r), (b_r) Tiles obtained after compression and reconstruction. (c), (d) Building mask obtained for test tile shown in (a) and (b). (c_r), (d_r) Building mask obtained for test tile shown in (a_r) and (b_r).

is reflected in the time required to implement the mathematical operations.

E. Results—Semantic Segmentation Application

The performance of the compression model is also evaluated in terms of the application of the reconstructed images. Particularly, a semantic segmentation application is considered in this case for the detection of buildings. The CAE model was validated by comparing the results obtained in the original images with those obtained with the decompressed images. For this purpose, a UNet model was trained using the Inria Aerial Image Labelling Dataset [45]. The dataset consists of aerial orthorectified RGB images with a spatial resolution of 0.3 m and the respective ground truth masks, which show the building depicted in the images. Images are acquired over different cities in the world. The dataset also includes unlabeled test images, which are used to predict building masks along with the corresponding images obtained after compression and reconstruction with the CAE model. The obtained masks are then compared to assess the differences between the two cases.

Inspired by [46], the UNet model includes five downsampling and five upsampling levels, with an increasing number of filters of 64, 128, 256, 512, 1024, and vice versa. The model was trained using binary cross-entropy loss function, Adam optimizer, and a decreasing learning rate starting from 10^{-5}. The early stopping algorithm was used to stop the training phase.

The dissimilarity is evaluated in terms of number of elements (buildings, in this case) that can be detected in the two masks, and in a pixel-based manner by calculating the hamming distance (HD) between the two masks. HD is defined as in (7) and varies in the range [0, 1] [47]. The dissimilarity between the mask obtained from the original image, denoted as I_1, and the mask obtained from the reconstructed image, denoted as I_2, is measured as in (7), where *i*, *j* are the coordinates, *M* and *N* are the width and height of the image

$$\mathrm{HD} = \frac{\sum_{i=1}^{M} \sum_{j=1}^{N} g\,(i, j)}{M \times N}. \tag{7}$$

$g(i, j)$ is defined as

$$g\ (i, j) = \begin{cases} 0, & I_1\ (i, j) = I_2\ (i, j) \\ 1, & I_1\,(i, j) \neq I_2\,(i, j)\,. \end{cases} \tag{8}$$

Fig. 16(a) and (b) shows two tiles with a dimension of 512 × 512 pixels extracted from two different test images, and Fig. 16(a_r) and (b_r) shows the corresponding one obtained after decompression. The predicted building masks obtained for the two original pairs of test tiles are shown in Fig. 16(c) and (d), while Fig. 16(c_r) and (d_r) shows the masks obtained from the reconstructed tiles. The predicted masks obtained for the original and reconstructed images are comparable. In both masks, the same number of objects can be identified, as summarized in Table VI and highlighted by the orange boxes in Fig. 16, and the HD computed as defined in (7) is equal to 0.016 and 0.014 for the tile reported in Fig. 16(c), (c_r), and the one reported in Fig. 16(d), (d_r), respectively.

The predicted masks are compared through the HD. As defined in (8), $g(i, j)$ is 0 if the pixels related to the two images located at i, j have the same value, 1 otherwise. The sum

TABLE VI
NUMBER OF IDENTIFIED OBJECTS IN TEST IMAGES IN FIG. 16

	Mask (original image)	Mask (reconstructed image)
Test case (a)	11	11
Test case (b)	20	20

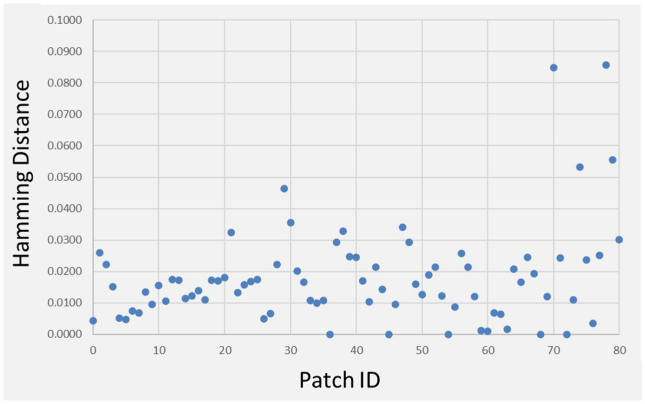

Fig. 17. HD between the predicted building masks.

of $g(i,j)$ over the whole image is normalized to the size of the total image, returning an error percentage. Fig. 17 reports the HD evaluated for each patch extracted from the test image. Values range between 0 and 0.085 approximately. This means that the predicted masks obtained from the original and reconstructed images differ in terms of pixels by about 8.5% at most.

IV. DISCUSSION

The algorithm here proposed aims at developing an on-board image compression method exclusively based on AI and compatible with a specific hardware. Therefore, several constraints have to be met, which can be summarized as follows:

1) The size of the encoder part of the model must be limited and compatible with the Myriad 2 processor in the framework of the Φ-Sat-2 mission, also considering that the on-board computer will run other applications. In addition, a small network will allow an easy update of the model, even in flight, under the uplink bandwidth limitations.
2) As explained in Section III-C, the model was converted through the OpenVINO toolkit in order to be run on the Myriad 2 VPU. Not all models or layers thar can be defined and developed with TensorFlow are supported by OpenVINO, therefore the developed NN model must be compatible with the OpenVINO version in use.
3) Encoder and decoder must operate as standalone modules, with the encoder working on-board and the decoder on the ground.
4) The final compression application that will be executed on the Φ-Sat-2 satellite must satisfy predefined requirements that include SSIM, CR, and processing time.

Our model is capable to meet such requirements.

For comparison with the other reviewed techniques, the only reference with an on-board hardware implementation is the one developed in [17]. In this case, the compression procedure proposed by the authors combines the CNN and JPEG compression and the hardware consists of a Zynq-7020 and a Zynq-MPSoC FPGA module. However, the encoder returns a compressed version of the input image that is half the size of the original one against the eight times obtained in this article. In [18], instead, CNN is used to obtain a compressed version of the input image, but the core of the compression is in another technique that is the NTD. In [19] and [20], both by the same authors, a CAE model is used but a dedicated analysis related to the use of the model for a specific hardware is not conducted. Moreover, the developed model can require a number of parameters in the encoder much higher than our model.

The encoder part of the CAE here presented has a size of 2.4 MB, that allows the model and model weights to be updated even during the in-orbit operation phase of the mission. Related to the performance achieved by the AI model, the results demonstrate the good quality of the reconstructed images. The average SSIM value is 0.96 with a CR of 8 for both the GPU and the VPU environments, and the images can be effectively used in application cases. Indeed, building detection test demonstrated that the same number of buildings can be identified using both the building mask obtained from the original image and the decompressed one. In addition, the results evaluated in terms of HD revealed that the two masks differ in terms of pixels of approximately 8.5% at most. Further developments can be focused on processing time optimization, which is quite higher on the VPU than on the GPU. This value should be lowered, in order to enable more massive on-board data acquisition. Furthermore, in order to achieve high compression rate, an intraband compression can be developed. The algorithm has not yet been tested on the final representative Φ-Sat-2 dataset, due to the lack of this new data. However, the accurate simulation of the Sentinel-2 acquisitions ensures a reliable evaluation of the results. In any case, the compression algorithm could be fine-tuned during the operational phase of the mission, when the acquisitions will be available.

V. CONCLUSION

The large amount of data produced by Earth Observation is a challenge today, and advanced processing on-board satellites can effectively help by extracting relevant information from data or by compressing it. In this article, the CAE-based image compression algorithm developed in the ESA Φ-Sat-2 CubeSat mission is presented. The mission aims at demonstrating the advantages and capabilities of running AI in space. The Myriad 2 VPU processor is used to run on-board a CubeSat several AI-based applications, and the method here described was developed also considering the constraints imposed by the hardware. The CAE is composed of an encoder that operates on-board the platform and reduces the size of the acquisition, and a decoder that operates on the ground and returns the reconstructed version of the input image. The CR, that is the ratio between the input image size and the compressed size, was determined by the dimension of the bottleneck layer of the CAE and was set equal to 8. The trained model was first tested on a GPU environment and then the performance achieved was compared with that obtained on the Myriad 2. The results were evaluated in terms of image quality metrics that are SSIM, PSNR, and MSE and using the reconstructed images in the applicative scenario of semantic segmentation. For all evaluation methods, we achieved good results, with an SSIM value around 0.96 for both the GPU and VPU computing platforms. Furthermore, we demonstrated that decompressed images can also be used for the semantic segmentation task, as they lead to results comparable to those obtained using the original images. A performance analysis of the model when operating in three different hardware environments has also been carried out. The processing time is the metric most affected by the migration from the GPU to the VPU environment, increasing from about 1 s to almost 11 s. Further developments will focus on the optimization of the processing time and also on studying interband compression to achieve a higher compression rate.

Acknowledgment

The Φ-Sat-2 project from the European Space Agency is led by Open Cosmos Ltd. We would like particularly to acknowledge the contribution of the team from the European Space Agency and from Open Cosmos Ltd. and all the consortium members: CGI, Ubotica, Simera CH Innovative, CEiiA, GEO-K, and KP Labs.

References

[1] B. Qi, H. Shi, Y. Zhuang, H. Chen, and L. Chen, “On-board, real-time preprocessing system for optical remote-sensing imagery,” *Sensors*, vol. 18, no. 5, 2018, Art. no. 1328.

[2] V. Kothari, E. Liberis, and N. D. Lane, “The final frontier: Deep learning in space,” in *Proc. 21st Int. Workshop Mobile Comput. Syst. Appl.*, 2020, pp. 45–49.

[3] D. Selva and D. Krejci, “A survey and assessment of the capabilities of CubeSats for earth observation,” *Acta Astronautica*, vol. 74, pp. 50–68, May/Jun. 2012.

[4] G. Giuffrida et al., “The Φ-sat-1 mission: The first on-board deep neural network demonstrator for satellite earth observation,” *IEEE Trans. Geosci. Remote Sens.*, vol. 60, Feb. 2022, Art. no. 5517414, doi: 10.1109/TGRS.2021.3125567.

[5] G. Giuffrida et al., “CloudScout: A deep neural network for on-board cloud detection on hyperspectral images,” *Remote Sens.*, vol. 12, no. 14, Jul. 2020, Art. no. 2205.

[6] 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.*, vol. 14, no. 13, 2022, Art. no. 3180.

[7] E. Rapuano et al., “An FPGA-based hardware accelerator for CNNs inference on board satellites: Benchmarking with myriad 2-based solution for the CloudScout case study,” *Remote Sens.*, vol. 13, no. 8, 2021, Art. no. 1518.

[8] M. P. Del Rosso, A. Sebastianelli, D. Spiller, P. P. Mathieu, and S. L. Ullo, “On-board volcanic eruption detection through CNNs and satellite multispectral imagery,” *Remote Sens.*, vol. 13, no. 17, 2021, Art. no. 3479.

[9] L. Diana, J. Xu, and L. Fanucci, “Oil spill identification from SAR images for low power embedded systems using CNN,” *Remote Sens.*, vol. 13, no. 18, 2021, Art. no. 3606.

[10] A. Hussain, A. Al-Fayadh, and N. Radi, “Image compression techniques: A survey in lossless and lossy algorithms,” *Neurocomputing*, vol. 300, pp. 44–69, 2018.

[11] H. M. Yasin and A. M. Abdulazeez, “Image compression based on deep learning: A review,” *Asian J. Res. Comput. Sci.*, vol. 8, no. 1, pp. 62–76, 2021.

[12] A. Altamimi and B. Ben Youssef, “A systematic review of hardware-accelerated compression of remotely sensed hyperspectral images,” *Sensors*, vol. 22, no. 1, 2022, Art. no. 263.

[13] J. Ballé, D. Minnen, S. Singh, S. J. Hwang, and N. Johnston, “Variational image compression with a scale hyperprior,” in *Proc. Int. Conf. Learn. Representations*, 2018, pp. 1–10.

[14] F. Jiang, W. Tao, S. Liu, J. Ren, X. Guo, and D. Zhao, “An end-to-end compression framework based on convolutional neural networks,” *IEEE Trans. Circuits Syst. Video Technol.*, vol. 28, no. 10, pp. 3007–3018, Oct. 2018.

[15] E. Agustsson, M. Tschannen, F. Mentzer, R. Timofte, and L. Van Gool, “Generative adversarial networks for extreme learned image compression,” in *Proc. IEEE/CVF Int. Conf. Comput. Vis.*, 2019, pp. 221–231.

[16] G. Toderici et al., “Full resolution image compression with recurrent neural networks,” in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2017, pp. 5435–5443.

[17] J. Goodwill, D. Wilson, S. Sabogal, A. D. George, and C. Wilson, “Adaptively lossy image compression for onboard processing,” in *Proc. IEEE Aerosp. Conf.*, 2020, pp. 1–15.

[18] J. Li and Z. Liu, “Multispectral transforms using convolution neural networks for remote sensing multispectral image compression,” *Remote Sens.*, vol. 11, no. 7, 2019, Art. no. 759.

[19] V. A. de Oliveira et al., “Reduced-complexity end-to-end variational autoencoder for on board satellite image compression,” *Remote Sens.*, vol. 13, no. 3, 2021, Art. no. 447.

[20] V. A. de Oliveira et al., “Satellite image compression and denoising with neural networks,” *IEEE Geosci. Remote Sens. Lett.*, vol. 19, Feb. 2022, Art. no. 4504105, doi: 10.1109/LGRS.2022.3145992.

[21] M. Pastena and N. Melega, “Overview of ESA earth observation CubeSats missions,” in *Proc. Sensors, Syst., Next-Gener. Satellites XXV*, 2021, doi: 10.1117/12.2597557.

[22] “Next artificial intelligence mission selected,” Sep. 2020, Accessed: Dec. 19, 2022. [Online]. Available: https://www.esa.int/Applications/Observing_the_Earth/Ph-sat/Next_artificial_intelligence_mission_selected

[23] C. Coelho, A. Marin, F. Deconinck, I. Babkina, J. Barrera Ars, and M. Pastena, “NanoSat MO framework: Enabling AI apps for earth observation,” in *Proc. Small Satell. Conf.*, 2021.

[24] A. Marin, C. Coelho, F. Deconinck, I. Babkina, N. Longepe, and M. Pastena, “Phi-Sat-2: Onboard AI apps for earth observation,” in *Proc. Space Artif. Intell.*, 2021.

[25] B. Sujitha, V. Murugan, L. Lydia, P. Rani, Z. Polkowski, and D. Shankar, “Optimal deep learning based image compression technique for data transmission on industrial internet of things applications,” *Trans. Emerg. Telecommun. Technol.*, vol. 32, 2021, Art. no. e3976.

[26] A. Hore and D. Ziou, “Image quality metrics: PSNR vs. SSIM,” in *Proc. 20th Int. Conf. Pattern Recognit.*, 2010, pp. 2366–2369.

[27] L. Fasano, D. Latini, A. Machidon, C. Clementini, G. Schiavon, and F. Del Frate, “SAR data fusion using nonlinear principal component analysis,” *IEEE Geosci. Remote Sens. Lett.*, vol. 17, no. 9, pp. 1543–1547, Sep. 2020.

[28] G. A. Licciardi and F. Del Frate, “Pixel unmixing in hyperspectral data by means of neural networks,” *IEEE Trans. Geosci. Remote Sens.*, vol. 49, no. 11, pp. 4163–4172, Nov. 2011.

[29] G. Guerrisi, F. Del Frate, and G. Schiavon, “Satellite on-board change detection via auto-associative neural networks,” *Remote Sens.*, vol. 14, no. 12, 2022, Art. no. 2735.

[30] M. Kramer, “Autoassociative neural networks,” *Comput. Chem. Eng.*, vol. 16, no. 4, pp. 313–328, Apr. 1992.

[31] Y. Zhang, “A better autoencoder for image: Convolutional autoencoder,” in *Proc. ICONIP17-DCEC*, 2018. Accessed: Sep. 5, 2023. [Online]. Available: http://users.cecs.anu.edu.au/~Tom.Gedeon/conf/ABCs2018/paper/ABCs2018_paper_58.pdf

[32] O. Yildirim, R. S. Tan, and U. R. Acharya, “An efficient compression of ECG signals using deep convolutional autoencoders,” *Cogn. Syst. Res.*, vol. 52, pp. 198–211, 2018.

[33] T. Liu, J. Wang, Q. Liu, S. Alibhai, T. Lu, and X. He, “High-ratio lossy compression: Exploring the autoencoder to compress scientific data,” *IEEE Trans. Big Data*, vol. 9, no. 1, pp. 22–36, Feb. 2023.

[34] Z. Wang, A. Bovik, H. Sheikh, and E. Simoncelli, “Image quality assessment: From error visibility to structural similarity,” *IEEE Trans. Image Process.*, vol. 13, no. 4, pp. 600–612, Apr. 2004.

[35] R. Dosselmann and X. D. Yang, “A comprehensive assessment of the structural similarity index,” *Signal, Image Video Process.*, vol. 5, pp. 81–91, 2011.

[36] U. Sara, M. Akter, and M. Shorif Uddin, “Image quality assessment through FSIM, SSIM, MSE and PSNR—A comparative study,” *J. Comput. Commun.*, vol. 7, pp. 8–18, 2019.

[37] N. Chervyakov, P. Lyakhov, and N. Nagornov, “Analysis of the quantization noise in discrete wavelet transform filters for 3D medical imaging,” *Appl. Sci.*, vol. 10, 2020, Art. no. 1223.

[38] J. Nay, “Single image super resolution using ESPCN—With SSIM loss,” *Available at SSRN 3898913*, 2021.

[39] D. S. Taubman and M. W. Marcellin, “JPEG2000: Standard for interactive imaging,” *Proc. IEEE*, vol. 90, no. 8, pp. 1336–1357, Aug. 2002.

[40] G. Furano et al., “Towards the use of artificial intelligence on the edge in space systems: Challenges and opportunities,” *IEEE Aerosp. Electron. Syst. Mag.*, vol. 35, no. 12, pp. 44–56, Dec. 2020.

[41] B. Barry et al., “Always-on vision processing unit for mobile applications,” *IEEE Micro*, vol. 35, no. 2, pp. 56–66, Mar./Apr. 2015.

[42] “Intel Movidius$^{\mathrm{TM}}$ vision processing units (VPUs),” Accessed: Jan. 3, 2023. [Online]. Available: https://www.intel.com/content/www/us/en/products/details/processors/movidius-vpu.html

[43] “OpenVINO,” Accessed: Jan. 17, 2023. [Online]. Available: https://docs.openvino.ai/latest/home.html

[44] A. Demidovskij et al., “OpenVINO deep learning workbench: A platform for model optimization, analysis and deployment,” in *Proc. IEEE 32nd Int. Conf. Tools Artif. Intell.*, 2020, pp. 661–668.

[45] E. Maggiori, Y. Tarabalka, G. Charpiat, and P. Alliez, “Can semantic labeling methods generalize to any city? The Inria aerial image labeling benchmark,” in *Proc. IEEE Int. Geosci. Remote Sens. Symp.*, 2017, pp. 3226–3229.

[46] O. Ronneberger, P. Fischer, and T. Brox, “U-Net: Convolutional networks for biomedical,” in *Proc. 18th Int. Conf. Med. Image Comput. Comput.-Assist. Intervention*, 2015, pp. 234–241.

[47] Z. Yang, L. Leng, and B.-G. Kim, “StoolNet for color classification of stool medical images,” *Electronics*, vol. 8, no. 12, 2019, Art. no. 1464.

Giorgia Guerrisi (Graduate Student Member, IEEE) received the M.Sc. degree (cum laude) in environmental engineering, in 2020, from the Tor Vergata University of Rome, Rome, Italy, where she is currently working toward the Ph.D. degree in computer science, control and geoinformation with a focus on the analysis of Earth remote sensing data using Artificial Intelligence.

In 2023, she was with the Remote Sensing Image Analysis Group, Technische Universität, Berlin, Germany. In particular, her research interest is in advanced satellite on-board processing operations with Deep Learning algorithms.

Fabio Del Frate (Senior Member, IEEE) received the laurea degree in electronic engineering and the Ph.D. degree in computer science from the Tor Vergata University, Rome, Italy, in 1992 and 1997, respectively.

From 1995 to 1996, he was a Visiting Scientist with the Research Laboratory of Electronics, Massachusetts Institute of Technology, Cambridge, MA, USA. In 1998 and 1999, he was with the ESRIN Centre, European Space Agency, Frascati, Italy. Since 1999, he has been with the University of Rome “Tor Vergata,” where he is currently a Full Professor, teaching courses on remote sensing and applied electromagnetism in various master’s and Ph.D. programs. In the same university, he is also the Coordinator of the “Design, Application, Regulation of UAVs” Master program and Erasmus Coordinator for the Engineering Macroarea. In the framework of remote sensing educational international activities, he has been a Lecturer in different European Universities (Prague, Trier, Cracow). He is, or has been, a Principal Investigator/Project Manager in several European Space Agency (ESA) and Italian Space Agency funded research projects. He has authored more than 200 international scientific publications, with a special focus on feature extraction algorithms from EO data using neural networks. In 2015, he was appointed by the European Organization for the Exploitation of Meteorological Satellites Associate Scientist for activities regarding the estimation of precipitation rate from satellite data. In 2019, he received an appointment by ESA as a Visiting Professor with the ESA ESRIN Center to provide support in the use of artificial intelligence for earth observation data processing. In 2006, he has cofounded GEO-K srl, the first spin-off company of the University of “Tor Vergata” of which he is currently the President. He has been an Associate Editor for the journal IEEE GEOSCIENCE AND REMOTE SENSING.

Giovanni Schiavon received the laurea degree (cum laude) in electronic engineering from the La Sapienza University, Rome, Italy, in 1982.

He has been a Full Professor (since 2012) with the Tor Vergata University, Rome, Italy, where he has been a Researcher since 1984, and an Associate Professor since 2000. He has been teaching courses on remote sensing since 1996, and on electromagnetic fields since 2000, and he has supervised about 15 Ph.D.s or postdocs. He has been a member of the board of directors of the Tor Vergata University, since 2021 (having held the same position from 2002 to 2005), where he has also been the Director of the Department of Civil Engineering and Computer Science Engineering from 2018 to 2021, and a member of the Academic Senate in the same period. He has been the Dean of the School of Engineering from 2014 to 2017. He has been the Coordinator of the Computer Science, Control and GeoInformation Ph.D. program from 2012 to 2017. He has managed several contracts with the European Space Agency and the Italian Space Agency. He has been involved in several international remote sensing projects since 1986. He has authored/coauthored more than 250 scientific papers, the most of which on international journals or proceedings with anonymous referees. His research interests include remote sensing of the atmosphere and of the earth’s surface, propagation, and millimeter waves.

Prof. Schiavon is a member of the Centro di Telerilevamento a Microonde and Società Italiana di Elettromagnetismo. He has served as a Reviewer for international journals, i.e., *Radio Science*, IEEE TRANSACTIONS ON GEOSCIENCE AND REMOTE SENSING, IEEE GEOSCIENCE AND REMOTE SENSING LETTERS, *Natural Hazards*, and for several volumes of the VSP publisher.

Open Access funding provided by ‘Universit? degli Studi di Roma "Tor Vergata"’ within the CRUI CARE Agreement