© 2026 IEEE. Personal use of this material is permitted. Permission from IEEE must be obtained for all other uses, in any current or future media, including reprinting/republishing this material for advertising or promotional purposes, creating new collective works, for resale or redistribution to servers or lists, or reuse of any copyrighted component of this work in other works.

This a **post-print** of the article “Exploring the Feasibility of Hailo 8L for Embedded Multispectral Classification” published in the Proceedings of IGARSS 2026 - 2026 IEEE International Geoscience and Remote Sensing Symposium.

The published article is available on https://doi.org/10.1109/IGARSS60577.2026.11730910

A. S. Garea, Á. Ordóñez, J. López-Fandiño, P. Quesada-Barriuso, F. Argüello and D. B. Heras, "Exploring the Feasibility of Hailo 8L for Embedded Multispectral Classification," *IGARSS 2026 - 2026 IEEE International Geoscience and Remote Sensing Symposium*, Washington DC, DC, USA, 2026, pp. 687-691, doi: 10.1109/IGARSS60577.2026.11730910.

© 2026 IEEE. Personal use of this material is permitted. Permission from IEEE must be obtained for all other uses, in any current or future media, including reprinting/republishing this material for advertising or promotional purposes, creating new collective works, for resale or redistribution to servers or lists, or reuse of any copyrighted component of this work in other works. The final publication is available at https://doi.org/10.1109/IGARSS60577.2026.11730910

Exploring the Feasibility of Hailo 8L for Embedded Multispectral Classification

Alberto S. Garea*†, Álvaro Ordóñez*†, Javier López-Fandiño*†, Pablo Quesada-Barriuso*†, Francisco Argüello†, Dora B. Heras*†

*Centro Singular de Investigación en Tecnoloxías Intelixentes (CiTIUS), Universidade de Santiago de Compostela, Spain.

†Departamento de Electrónica e Computación, Universidade de Santiago de Compostela, Spain

{jorge.suarez.garea, alvaro.ordonez, javier.lopez.fandino, pablo.quesada, francisco.arguello, dora.blanco}@usc.es

***Abstract*—Multispectral remote sensing image classification typically imposes high computational demands, limiting its use for real-time applications on resource-constrained platforms. This work presents an embedded AI framework for efficient remote sensing image classification using a Raspberry Pi 5 equipped with a Hailo-8L AI chip accelerator. Two deep learning architectures, a lightweight CNN and a more complex GAN-based ResBaGAN model, were adapted and optimized for execution in this framework, achieving significant inference speedups while maintaining competitive accuracy. Experiments conducted on multispectral images demonstrate that Hailo-8L enables the Raspberry Pi to achieve execution times comparable to those of a more powerful desktop device. The results confirm the feasibility of deploying sophisticated remote sensing classification models on low-power embedded systems, opening new possibilities for on-site, cost-effective environmental monitoring applications.**

***Index Terms*—Raspberry Pi, hardware accelerator, multispectral, remote sensing, classification, deep learning, embedded computing.**

I. Introduction

One of the main advantages of remote sensing is its ability to monitor a wide range of areas over long distances. The combination of drone mobility with remote sensing (RS) technologies has fostered the growth of numerous drone-based RS applications [1]. For large regions such as river basins, urban environments, or croplands, drones equipped with RS sensors can gather extensive amounts of multispectral data within a short time period, reaching several gigabytes per minute and making real-time transmission particularly challenging.

Frequently, data are stored in the drone's internal memory and then transferred to ground-based workstations equipped with Graphics Processing Units (GPU) for offline post-flight analysis [2]. However, this approach cannot meet the needs of applications that rely on immediate results, which would require performing the processing directly on board the drone. Implementing such real-time analysis is further complicated by the need for advanced hardware accelerators like GPUs, which are difficult to integrate into aerial platforms due to limitations in size, weight, and power consumption.

This work was supported by project PID2022-141623NB-I00 funded by MCIN/AEI/10.13039/501100011033, and by Xunta de Galicia – Consellería de Educación, Ciencia, Universidades e Formación Profesional (Centro de investigación de Galicia accreditation 2024-2027 ED431G-2023/04 and Reference Competitive Group accreditation ED431C-2022/16), both co-funded by ERDF/EU.

These constraints highlight the challenge of designing or adapting methods for real-time RS image classification on embedded platforms, providing on-site results at a significantly lower cost compared to conventional computing solutions. Compact, low-power embedded systems such as the Raspberry Pi are engineered to execute specific tasks efficiently and cost-effectively. This paradigm aligns with the principles of edge computing (EC), which advocates bringing computation closer to data sources to improve latency and bandwidth utilization.

EC and drone-based systems intersect in a way that enables the latter to deliver computational services closer to end users by processing data locally, thereby eliminating the need to transmit information through network infrastructures to centralized data centers. The efficiency and responsiveness provided by these systems are fostering the integration of artificial intelligence (AI) into EC devices [3]. Among these advances are dedicated AI accelerator chips, such as the Hailo-8L, which supports real-time inference with low energy consumption and reduced cost.

These developments naturally raise the question of whether large deep learning (DL) models, typically used for RS image classification, can be executed on low-power embedded platforms with limited computational capacity. This enables remote sensing applications capable of acquiring and processing data in real-time directly on site.

This article presents an evaluation of the performance of the low-power embedded Hailo-8L accelerator chip for some adapted multispectral remote sensing image classification models. The obtained classification results and inference times are compared with those achieved on desktop devices.

II. Hardware devices for edge computing

The EC paradigm has emerged in response to the growing use of Internet of Things (IoT) devices, which, despite their limited computational capacity and memory, achieve high energy efficiency. While cloud computing remains a powerful option for processing large-scale data, network latency and bandwidth limitations make it unfeasible for time-critical tasks such as real-time environmental monitoring. This motivates the development of architectures capable of performing inference near the data source, without relying on remote servers [4].

© 2026 IEEE. Personal use of this material is permitted. Permission from IEEE must be obtained for all other uses, in any current or future media, including reprinting/republishing this material for advertising or promotional purposes, creating new collective works, for resale or redistribution to servers or lists, or reuse of any copyrighted component of this work in other works. The final publication is available at https://doi.org/10.1109/IGARSS60577.2026.11730910

However, the limited hardware resources of EC devices constrain the deployment of complex machine learning models. A common compromise is to perform preliminary data processing locally while transmitting the most relevant features to external servers for further analysis. In this context, EC refers to executing computations as close as possible to the point of acquisition, minimizing latency and dependence on remote infrastructures.

To meet these processing requirements, different architectural strategies have been proposed: on-device computing, which runs deep neural networks (DNN) directly on the device [5]; edge server–based computing [6], which offloads inference to nearby servers; and joint computing [7], which distributes the execution between device and server.

This article focuses on on-device computing, as it aligns with the hypothesis of performing inference directly on embedded systems. Deploying AI models on such devices requires careful selection of compatible hardware and model compression techniques [8]. Developers typically design lightweight architectures with fewer parameters to reduce memory footprint and inference time while maintaining reliable accuracy.

The primary strategy for enabling AI inference on embedded devices is model compression, achieved mainly through quantization and pruning.

- **Pruning** eliminates redundant parameters, such as weak connections or near-zero weights, to reduce model complexity and improve efficiency while preserving performance. A more flexible variant, soft pruning, constrains weight values instead of removing them completely, enhancing the model's ability to generalize.
- **Quantization** reduces model precision by converting 32-bit or 16-bit floating-point weights to lower-precision integers (e.g., 8-bit or 4-bit). This technique decreases computational cost and energy consumption with minimal accuracy degradation. Because model weights are conventionally stored in 32-bit format, quantization can reduce model size by up to a factor of four.

Selecting appropriate hardware is essential for achieving an optimal balance among energy efficiency, cost, latency, and accuracy. While desktop systems or servers could, in principle, act as edge devices, their size and energy demands are prohibitive in most on-board scenarios. For this reason, dedicated edge AI hardware has been developed to support industrial and embedded applications. Typical hardware solutions include:

- **AI accelerators:** Chips optimized for vector and tensor operations, essential for neural networks. Examples include Intel's Neural Compute Engine, supporting FP16 precision and integration with frameworks such as Caffe and TensorFlow. These accelerators enable real-time inference with low latency and high energy efficiency on edge devices.
- **Hailo-8L:** An AI accelerator supporting 8 and 16-bit operations within Open Neural Network Exchange (ONNX) and TensorFlow frameworks. The Hailo family of accelerators, selected for this article, integrates the AI chip, memory, power management, and interface in a single module designed to support products that require limited AI capabilities or low-performance computing (see Fig. 1). A quantization strategy using 8-bit precision by default enables models execution on this AI accelerator. An optional preceding equalization step scales weight ranges, and implicitly activation ranges, across different layers or channels in order to reduce subsequent quantization errors [9].
- **Field-Programmable Gate Arrays (FPGAs):** Reconfigurable hardware that can be customized for specific AI tasks, such as the MAX V CPLD or Cyclone 10 LP/GX FPGAs, frequently used in edge AI designs.
- **Google Coral [10]:** A complete computing platform based on the Edge Tensor Processing Units (TPUs), capable of delivering up to 4 tera-operations per second (TOPS) and supporting connections to cameras and sensors, optimized for deployment in production environments.
- **NVIDIA Jetson [11]:** A family of embedded computing modules based on a system-on-a-chip (SoC) architecture integrating ARM CPUs and Nvidia GPUs. Designed for energy-efficient machine learning acceleration, Jetson includes the JetPack SDK and a comprehensive software ecosystem, making it one of the most widely adopted solutions for edge AI development.

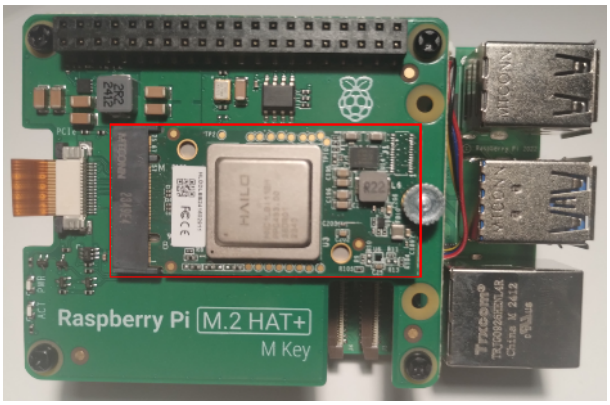


Fig. 1. Hailo-8L module (red rectangle) connected to a Raspberry Pi 5 via a PCIe 2.0 interface (M.2 HAT+)

III. Deep Learning for Remote Sensing Image Classification

Traditional machine learning classifiers, such as support vector machines (SVMs) and extreme learning machines (ELMs), have largely been superseded by DL frameworks capable of jointly learning multiple levels of representation to extract more meaningful and discriminative features from the data [12]. One of the most widely used approaches for RS image classification is convolutional neural networks (CNNs). CNNs efficiently learn image representations by applying convolutional filters and hierarchical structures. A standard

© 2026 IEEE. Personal use of this material is permitted. Permission from IEEE must be obtained for all other uses, in any current or future media, including reprinting/republishing this material for advertising or promotional purposes, creating new collective works, for resale or redistribution to servers or lists, or reuse of any copyrighted component of this work in other works. The final publication is available at https://doi.org/10.1109/IGARSS60577.2026.11730910

CNN architecture comprises convolutional layers, activation functions (such as the rectified linear unit), pooling layers, and fully connected (FC) layers [13].

Another DL paradigm explored in RS classification is generative adversarial networks (GANs). While supervised learning models like CNNs achieve high classification accuracy, they usually require large labeled datasets. In RS, however, labeled data are scarce due to the cost and labor involved in annotation, which often leads to overfitting, i.e., strong performance on the training data but poor generalization. To address this limitation, GAN-based training strategies have been proposed. A GAN consists of two competing models: a generator and a discriminator. The generator creates synthetic samples designed to resemble real ones, while the discriminator learns to distinguish genuine data from generated data. Through their adversarial interaction, both models progressively improve, resulting in increasingly realistic synthetic data and more accurate discrimination. Both approaches, based on CNNs and GANs, make use of DNNs that require devices such as GPUs for faster processing.

IV. Results

A dataset covering a region of the Oitavén river watershed[1] (hereafter referred to as Oitavén) was used for the experiments [14]. Data were acquired by a MicaSense RedEdge multispectral camera mounted on a custom unmanned aerial vehicle (UAV). This sensor captures five spectral bands: blue (475 nm), green (560 nm), red (668 nm), red-edge (717 nm), and near-infrared (NIR, 840 nm). The image, shown in Fig. 2 as a false-color composite, contains a total of 22,041,591 data values, of which 6,067,179 pixels are labeled. It was captured at a flight altitude of 120m, with a spatial resolution of 10cm/pixel, during the summer of 2018 in Galicia, Spain. It has a spatial size of 6,689×6,722 pixels. Ten land-cover classes were considered for the experiments. The corresponding reference data used for classification are shown in Fig. 3, while Table I presents each class along with its representative color, name, and number of available samples.

TABLE I
Reference data description

Class	**Unknown**	**Water**	**Bare Soil**	**Rock**
Pixels	38,896,279	309,248	113,324	79,152
Class	**Asphalt**	**Concrete**	**Tiles**	**Meadows**
Pixels	43,861	128,022	78,785	2,428,482
Class	**Native trees**	**Pines**	**Eucalyptus**	
Pixels	1,829,360	193,884	863,061	

Two different hardware platforms are compared in this article. On one side, a laptop equipped with an eight-core AMD Ryzen 7 5800H CPU at 3.20 GHz and 16 GB of RAM. On the other side, a Raspberry Pi 5 featuring a quad-core ARM Cortex-A76 processor at 2.4 GHz and 8 GB of RAM. A Hailo-8L AI Accelerator, providing up to 13 TOPS, was connected to the Raspberry Pi via a PCIe 2.0 ×1 interface. To deploy the models on the Hailo-8L chip, the original network must first be converted into ONNX format. This conversion was performed using the Hailo AI Software Suite [15]. Regarding the operating system, a 64-bit Debian distribution (Bookworm) was used on both the laptop and the Raspberry Pi.

[1]This image was obtained as part of the Civil Program UAVs Initiative, promoted by the Xunta de Galicia.

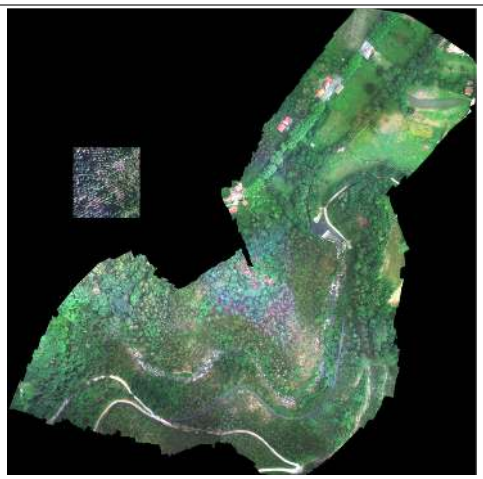


Fig. 2. Oitavén river false-color composite

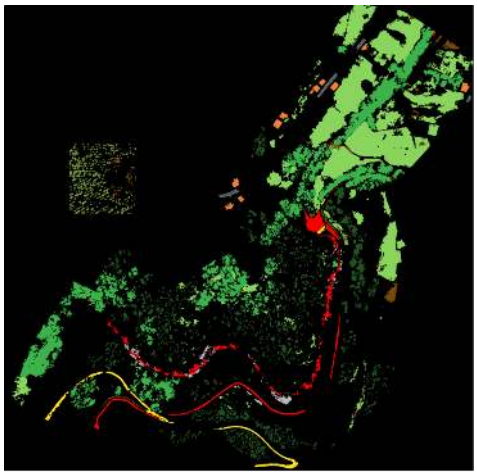


Fig. 3. Oitavén river reference data

Two DL architectures were employed in the experiments: a CNN and a GAN. The CNN architecture consisted of two convolutional layers, each followed by a max-pooling layer, and a final classification stage implemented as a fully connected layer. Overall, the network contained 34,154 trainable parameters. During training, the labeled dataset was split into 2% for training, 1% for validation, and 97% for testing. The model was trained for 100 iterations using a batch size of 100 samples and a learning rate of 0.001. The Adam optimizer was adopted to update model weights, while the cross-entropy loss function was used to evaluate prediction errors during learning.

In the second case, the residual balancing GAN (ResBaGAN) proposed in [16] was employed. This network is based on a GAN architecture enriched with autoencoder-based

© 2026 IEEE. Personal use of this material is permitted. Permission from IEEE must be obtained for all other uses, in any current or future media, including reprinting/republishing this material for advertising or promotional purposes, creating new collective works, for resale or redistribution to servers or lists, or reuse of any copyrighted component of this work in other works. The final publication is available at https://doi.org/10.1109/IGARSS60577.2026.11730910

initialization and class-balancing mechanisms, a superpixel-driven sample extraction strategy combined with standard data augmentation techniques, and an improved residual network used as the classifier. The same distribution of labeled training, validation and test samples as in the CNN case was used for training. A dropout probability of 0.05 and the Xavier method for weight initialization were applied. In total, the network consists of 941,371 parameters.

To evaluate the performance of the models, the selected metrics were inference time, overall accuracy (OA), and average accuracy (AA), as the assessment focuses on both the required execution time and the accuracy achieved after applying different optimization techniques.

Table II shows the inference times (in seconds) for both the CNN and the ResBaGAN models across the three selected configurations: the PC configuration referring to the model executed on the laptop CPU, the edge device (ED) configuration corresponding to the model executed on the Raspberry Pi CPU, and the Accelerated ED (AED) referring to the model executed on the Raspberry Pi together with the Hailo-8L accelerator.

TABLE II
INFERENCE TIMES

Network	Configuration	Hardware	Data Type	Time (s)
CNN	PC	AMD Ryzen 7 5800H	32 bits	476.48
	ED	ARM Cortex-A76	32 bits	3444.69
	AED	ARM + Hailo-8L	8 bits	666.16
ResBaGAN	PC	AMD Ryzen 7 5800H	32 bits	1518.88
	ED	ARM Cortex-A76	32 bits	11 259.37
	AED	ARM + Hailo-8L	8 bits	988.36

The inference results highlight significant differences in performance across hardware platforms on both networks. For the CNN, execution on the PC configuration achieves 476.48 s, while the ED configuration requires 3,444.69 s due to the embedded platform's limitations. The AED configuration demonstrates that hardware acceleration significantly improves performance, reducing the time to 666.16 s. In this case, the inference time is close to that obtained on the laptop, but the small model size limits the benefits of both applied optimizations and hardware accelerator, preventing substantial improvements. Similarly, the ResBaGAN on the PC configuration requires substantially less time (1,518.88 s) compared to the ED configuration (11,259.37 s). The larger model size of ResBAGAN allows the AED configuration to obtain a more significant reduction in inference time (988.36 s), even improving the PC configuration, thus demonstrating the effectiveness of dedicated hardware acceleration for enabling these models in ED scenarios.

Regarding classification accuracy, Table III presents the OA and AA values for the different configurations. It can be seen that the ED configuration achieves the same discrimination performance as the PC configuration, while the AED configuration, as expected, loses some accuracy in exchange for reduced execution times. Nevertheless, as it can be seen in Table IV, these losses are small for all classes in the CNN model and concentrate in particular for the ResBaGAN problem for the classes with the lowest number of samples.

TABLE III
CLASSIFICATION ACCURACY

Network	Configuration	OA (%)	AA (%)
CNN	PC	87.95	85.88
	ED	87.95	85.88
	AED	86.93	84.53
ResBaGAN	PC	97.17	95.82
	ED	97.17	95.82
	AED	81.82	66.09

TABLE IV
CLASSIFICATION ACCURACIES PER CLASS

	CNN			ResBaGAN		
Class	PC (%)	ED (%)	AED (%)	PC (%)	ED (%)	AED (%)
Water	97.94	97.94	98.07	99.01	99.01	98.69
Bare Soil	91.47	91.47	92.39	91.91	91.91	51.64
Rock	60.19	60.19	57.06	87.23	87.23	0.01
Asphalt	98.62	98.62	98.57	97.64	97.64	45.23
Concrete	79.71	79.71	78.43	91.87	91.87	60.89
Tiles	89.18	89.18	82.36	99.20	99.20	57.29
Meadows	97.74	97.74	97.96	96.65	96.65	97.58
Native Trees	85.81	85.81	82.11	98.35	98.35	59.52
Pines	96.02	96.02	94.90	99.81	99.81	95.87
Eucalyptus	62.18	62.18	63.41	96.23	96.23	94.18

V. CONCLUSIONS

This article addresses the problem of remote sensing image classification with DL deployed on EC platforms. The choice of this setting is driven by the benefits of EC for real-time applications in remote environments, where models can run without dependence on large-scale data centers, leading to lower operational costs and reduced energy consumption. The experiments conducted demonstrated that employing the Hailo-8L accelerator chip yields a decrease in inference times, with model size directly influencing the magnitude of the performance gains achieved, while maintaining competitive accuracy.

Future work will evaluate more complex DL architectures to fully exploit the capabilities of the Hailo 8L. Furthermore, we plan to extend these experiments to additional EC platforms.

REFERENCES

[1] Z. Zhang and L. Zhu, "A review on unmanned aerial vehicle remote sensing: Platforms, sensors, data processing methods, and applications," *drones*, vol. 7, no. 6, p. 398, 2023.

[2] B. Kim, J. Jung, H. Min, and J. Heo, "Energy efficient and real-time remote sensing in ai-powered drone," *Mobile Information Systems*, vol. 2021, no. 1, p. 6650053, 2021.

[3] P. McEnroe, S. Wang, and M. Liyanage, "A survey on the convergence of edge computing and ai for uavs: Opportunities and challenges," *IEEE Internet of Things Journal*, vol. 9, no. 17, pp. 15 435–15 459, 2022.

[4] X. Kong, Y. Wu, H. Wang, and F. Xia, "Edge computing for internet of everything: A survey," *IEEE internet of things journal*, vol. 9, no. 23, pp. 23 472–23 485, 2022.

© 2026 IEEE. Personal use of this material is permitted. Permission from IEEE must be obtained for all other uses, in any current or future media, including reprinting/republishing this material for advertising or promotional purposes, creating new collective works, for resale or redistribution to servers or lists, or reuse of any copyrighted component of this work in other works. The final publication is available at https://doi.org/10.1109/IGARSS60577.2026.11730910

[5] S. Zhu, T. Voigt, F. Rahimian, and J. Ko, "On-device training: A first overview on existing systems," *ACM transactions on sensor networks*, vol. 20, no. 6, pp. 1–39, 2024.

[6] A. Lu, R. Cao, Y. Wang, W. Hu, Y. Gao, Z. Hu, and Y. Zang, "Edge computing and server-based high-precision flood level classification system," *Engineering Applications of Artificial Intelligence*, vol. 162, p. 112442, 2025.

[7] S. Zhang, G. Cui, Y. Long, and W. Wang, "Joint computing and communication resource allocation for satellite communication networks with edge computing," *China Communications*, vol. 18, no. 7, pp. 236–252, 2021.

[8] Z. Zhang and J. Li, "A review of artificial intelligence in embedded systems," *Micromachines*, vol. 14, no. 5, p. 897, 2023.

[9] A. Gholami, S. Kim, Z. Dong, Z. Yao, M. W. Mahoney, and K. Keutzer, "A survey of quantization methods for efficient neural network inference," in *Low-power computer vision*. Chapman and Hall/CRC, 2022, pp. 291–326.

[10] G. LLC, "Google coral," 2025, https://www.coral.ai/ [Accessed: 2025/12/12].

[11] N. Corporation, "Nvidia jetson," 2025, https://www.nvidia.com/es-es/autonomous-machines/embedded-systems/ [Accessed: 2025/12/12].

[12] B. Hosseiny, M. Mahdianpari, M. Hemati, A. Radman, F. Mohammadimanesh, and J. Chanussot, "Beyond supervised learning in remote sensing: A systematic review of deep learning approaches," *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, vol. 17, pp. 1035–1052, 2023.

[13] D. Dai, "An introduction of cnn: models and training on neural network models," in *2021 International Conference on Big Data, Artificial Intelligence and Risk Management (ICBAR)*. IEEE, 2021, pp. 135–138.

[14] F. Argüello, D. B. Heras, A. S. Garea, and P. Quesada-Barriuso, "Watershed monitoring in galicia from uav multispectral imagery using advanced texture methods," *Remote Sensing*, vol. 13, no. 14, p. 2687, 2021.

[15] H. T. LTD, "Hailo ai sotware suite," 2025, https://hailo.ai/products/hailo-software/hailo-ai-software-suite/ [Accessed: 2025/12/12].

[16] Á. G. Dieste, F. Argüello, and D. B. Heras, "Resbagan: A residual balancing gan with data augmentation for forest mapping," *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, vol. 16, pp. 6428–6447, 2023.