





# A GPU Accelerated Hyperspectral 3D Convolutional Neural Network Classification at the Edge with Principal Component Analysis Preprocessing

Gianluca De Lucia<sup>1</sup>✉ , Marco Lapegna<sup>2</sup> , and Diego Romano<sup>1</sup> 

<sup>1</sup> Institute for High Performance Computing and Networking (ICAR), CNR,  
80131 Naples, Italy

{gianluca.delucia,diego.romano}@icar.cnr.it

<sup>2</sup> Department of Mathematics and Applications, University of Naples Federico II,  
80126 Naples, Italy

marco.lapegna@unina.it

**Abstract.** The Edge Computing paradigm promises to transfer decision-making processes based on artificial intelligence algorithms to the edge of the network without the need to query servers far from the data collection point. Hyperspectral image classification is one of the application fields that can benefit most from the close relationship between Edge Computing and Artificial Intelligence. It consists of a framework of techniques and methodologies for collecting and processing images related to objects or scenes on the Earth's surface, employing cameras or other sensors mounted on Unmanned Aerial Vehicles. However, the computing performance of the edge devices is not comparable with those of high-end servers, so specific approaches are required to consider the influence of the computing environment on the algorithm development methodology. In the present work, we propose a hybrid technique to make the Hyperspectral Image classification through Convolutional Neural Network affordable on low-power and high-performance sensor devices. We first use the Principal Component Analysis to filter insignificant wavelengths to reduce the dataset dimension; then, we use a process acceleration strategy to improve the performance by introducing a GPU-based form of parallelism.

**Keywords:** Hyperspectral classification · Edge Computing · Principal Component Analysis · GPU computing

## 1 Introduction

Edge computing refers to the enabling technologies to process data at the network's edge near the data source before being sent to the cloud data center. For some authors, edge computing is interchangeable with fog computing [1], although it focuses more on the devices at the edge, whereas fog computing focuses more on the whole network infrastructure. This type of computing paradigm has several advantages over traditional cloud computing, e.g., as the

results of [15] show, energy savings can reach up to 40%. There are various metrics to consider in Edge computing, including energy and transmission rate, especially for big data [3, 13]. In addition to the network signal strength [11], data size and available bandwidth will also influence the transmission energy overhead [27]. For this reason, as shown in [17], new High-Performance Edge devices mount GPUs capable of performing complex calculations by finding a trade-off between performance and power consumption.

One stimulating application field for Edge computing is Remote Sensing (RS). RS is the science of acquiring, processing, and interpreting images and related data from aircraft and satellites that record the interaction between matter and electromagnetic energy [30]. In recent years, deep learning techniques have revolutionized how RS images are processed and classified. In particular, standard optical, RGB, and IR (infrared) images have benefited from deep convolutional neural networks (CNNs) for classification, object detection, or semantic segmentation tasks [6, 25, 33].

A promising RS technology focuses on hyperspectral images (HSIs), allowing simultaneous radiance capture at different wavelengths, and generating various spectral bands. HSI data have an exceptionally high range and resolution in the spectral dimension. In particular, the branch of Hyperspectral Imaging deals with collecting and processing information on the nature of materials by analyzing their reflectance in a part of the electromagnetic spectrum [12]. Hyperspectral imaging aims to obtain a spectral vector for each pixel of an image to find objects, detect processes, or identify and classify materials [8, 10].

Some classifiers preprocess the HSI to reduce the image depth to three spectral bands (RGB) through Principal Component Analysis (PCA) or other strategies [23, 31] and only use a 2D CNN architecture to perform the classification. However, this approach may result in the loss of some hyperspectral properties. For this reason, we propose to use PCA to reduce the length of the HSI spectral dimension while maintaining the multidimensional nature of the data. This strategy allows adoption of more accurate and faster classification tools than the above methods.

In this paper, we will present an HSI classifier<sup>1</sup> that exploits the computational power of the GPU on High-Performance Edge Devices. For the development, we used a PyTorch-based deep learning toolbox for classifying hyperspectral data called DeepHyperX [4]. We focused on three-dimensional convolutional networks (3D CNNs). Indeed, since we can interpret HSIs as volumes, we can classify them with the aid of 3D CNNs using three-dimensional convolutions [20]. Instead of producing 2D feature maps, these 3D CNNs create 3D feature maps suitable for spectral pattern recognition and seem theoretically more relevant for HSI classification. This approach slightly improves classification performance compared to 2D+1D models [19]. In [9], the author showed that 3D CNNs for the classification of hyperspectral images performed better than their 2D counterparts. Indeed, compared to spectral CNNs or 2D+1D counterparts, 3D CNNs combine spatial and spectral pattern recognition strategies in one filter, requiring fewer parameters and layers.

---

<sup>1</sup> Source code: <https://github.com/gigernau/PCAHyperspectralClassifier>.

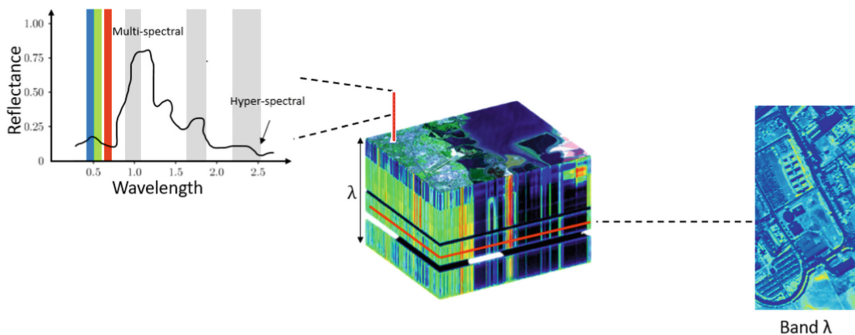
Many architectures in the literature handle 3D convolutional neural networks for hyperspectral data [7, 9, 16, 18, 21, 22]. The authors of [5] compare several variants, pointing out their ability to recognize more complex 3D reflectance patterns, such as spectral signatures and absorption differences between bands.

With this work, we want to show how High-Performance Edge Computing can enable onboard classification with a limited energetic impact, eventually improving the transmission stage towards the ground station. The idea is to preprocess raw data using a GPU-parallel PCA to reduce the spectral dimension of the HSI while retaining the information content. Then, a properly chosen GPU-accelerated 3D CNN classifier [20] can process the hyperspectral-reduced data in a shorter time while maintaining high accuracy.

## 2 HSI Pipeline

HSIs have a data structure similar to RGB images, consisting of the superposition of three wavelengths, one for each primary color: red, green, and blue. Even if the visible spectrum has a broader range of wavelengths, RGB images appear to the human eye in almost any color, thanks to the tristimulus mechanism. In hyperspectral cameras, images have higher information content. HSI cameras allow the simultaneous capture of radiance at different wavelength bands of the electromagnetic spectrum, providing informative spectral details for each material. An HSI has spatial pixels corresponding to geographical locations, each with a spectral depth of several wavelength bands depending on the specific sensor. Thus, an HSI is a volume graphically representable with a so-called cube of hyperspectral data (Fig. 1).

If we cut the cube perpendicularly to the spectral bands, we obtain a plane appearing as an image whose pixels represent the reflectance at a specific wavelength  $\lambda$ . Therefore, the pixel's intensity, with a value usually normalized between 0 and 1, measures the surface efficiency of the sampled material in radiative reflection at  $\lambda$ .



**Fig. 1.** Graphical representation of a hyperspectral data cube.

Two main methods of reducing datasets are PCA and Multidimensional Scaling (MDS). We preferred to focus on using PCA, which operates on the spectral dimension, rather than MDS. The output of the HSI classification produces labels for each pixel, so we have to preserve the spatial details, while MDS focuses mainly on reducing the spatial dimensions.

Thanks to PCA, we can reduce the spectral dimension by projecting the vector corresponding to each spatial point onto the first principal components only, where the variance of the data and the information content are most relevant. We can define the first principal component as the direction that maximizes the variance of the projected data. The  $i$ -th principal component is the direction orthogonal to the first  $i - 1$  principal components that maximize the variance of the projected data [28]. The main steps of PCA are [32]:

- Dataset normalization.
- Calculating covariance matrix for the features in the dataset.
- Calculating eigenvalues and eigenvectors for the covariance matrix.
- Ordering eigenvalues and corresponding eigenvectors.
- Selection of  $k$  eigenvalues and creation of the eigenvectors matrix.

The eigenvector associated with the largest eigenvalue indicates the direction in which the data have the greatest variance.

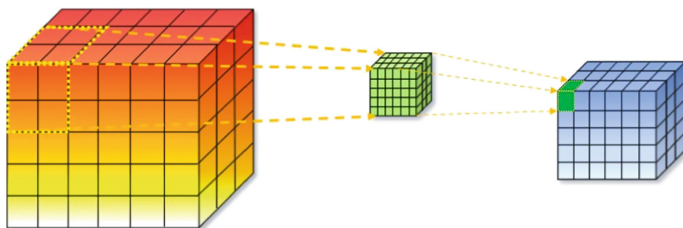
In general, dimensionality reduction inevitably results in a loss of information, leading to less accurate data classification. However, PCA minimizes this information loss. Moreover, available parallel implementations on SIMD architectures can exploit GPU acceleration using a SIMT execution model [29]. Indeed, optimized versions of GPU-parallel cuBLAS-based PCA are up to 12 times faster than the CPU-optimised BLAS versions [2]. Our high-performance PCA cuBLAS implementation uses the Gram-Schmidt orthogonalization, as described in [2]. Therefore, we will perform PCA on the dataset before the classification phase to speed up the process without sacrificing prediction accuracy.

For HSI classification through Deep Learning, many authors use CNNs [5]. In general, classifiers built with CNNs usually have the following layers:

- **Convolutional layers:** filters extract the features of the images analyzed.
- **Pooling layers:** reduce the dimension of the feature maps by downsampling, and increase the level of *abstraction*.
- **Fully Connected layers:** work as traditional feed-forward neural networks, in which all neurons connect to all neurons from the previous layer.
- **Output layer:** a fully connected layer using *softmax* as a trigger function to obtain the selected input’s probabilities for a specific class.

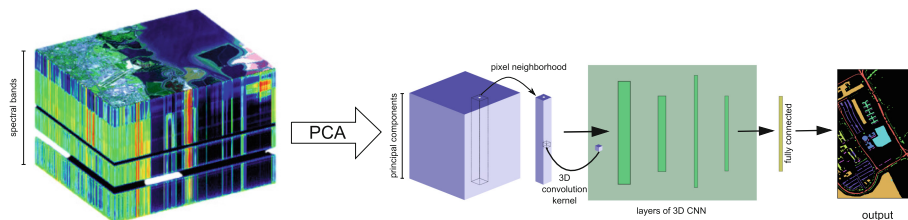
In a fully connected layer, an **activation function** computes the weighted sum of neurons of the previous layer and consequently activates neurons on the current layer. In particular, the Rectified Linear Units (ReLU) function has excellent performance on deep networks; therefore, many authors currently prefer it.

We will use a 3D-CNN, where the filters used in the convolutional layers are three-dimensional and move along the three directions to calculate feature representations (Fig. 2).



**Fig. 2.** Example of three-dimensional convolution.

Hence, the pipeline of our classifier (Fig. 3) takes hyperspectral data as input, then performs a GPU-parallel PCA by executing the code in CUDA. Next, the reduced dataset becomes the input for the inference via the appropriately trained 3D-CNN network model. The output is an RGB image in which each pixel has a color representing the class of the corresponding material.



**Fig. 3.** Pipeline of the HSI classifier with PCA preprocessing.

### 3 Experiments

We developed a hyperspectral image classifier trained on two datasets to test our approach. There are few public datasets [14] acquired using hyperspectral sensors. In particular, for this work, we used:

- **Indian Pines (IP):** collected by the AVIRIS sensor on a NASA flight over northwestern Indiana in 1992, with a ground pixel resolution of 17 m. The acquired data consist of  $145 \times 145$  pixels with 220 spectral bands, but after removing the water absorption bands (104 – 108, 150 – 163, and 220), they result in 200 bands. The ground truth has 16 classes, not all of which are mutually exclusive.
- **Pavia University (PU):** detected by the ROSIS sensor on a DLR flight in 2002 over Pavia, Italy, with a ground pixel resolution of 1.3 m. After removing samples without information, the dataset consists of  $610 \times 340$  pixels, with 103 spectral bands. The ground truth differentiates 9 classes.

We used double precision for both datasets to present coherent results during our tests, even if the original formats differed.

We experimented on two different platforms:

- PC with a 2.60 GHz Intel Core i7-9750H CPU, 16 GB RAM, Nvidia GeForce RTX 2060 GPU, and running Ubuntu Linux;
- Nvidia Jetson Nano developer kit.

We exploited the GPUs on both platforms to accelerate each step of the classification pipeline. To test our code in the High-Performance Edge Computing environment, we used the Jetson Nano activating both 5W and 10W modalities and reporting their impact on the inference time and the energy absorption.

Firstly, we selected the best 3D-CNN model in inference time and prediction accuracy for both datasets on the Jetson Nano. This step is essential to identifying the most promising model for the Edge computing environment. Then, we tested the overall classification performance in prediction accuracy and inference time by changing the selected number of components in the PCA preprocessing. We used a customized parallel version of the PCA developed in CUDA using the cuBLAS library for this task. We also compared the execution time of the PCA preprocessing using our CUDA version and the scikit-learn module of Python. Finally, we evaluated the energy consumption using both Jetson Nano modalities.

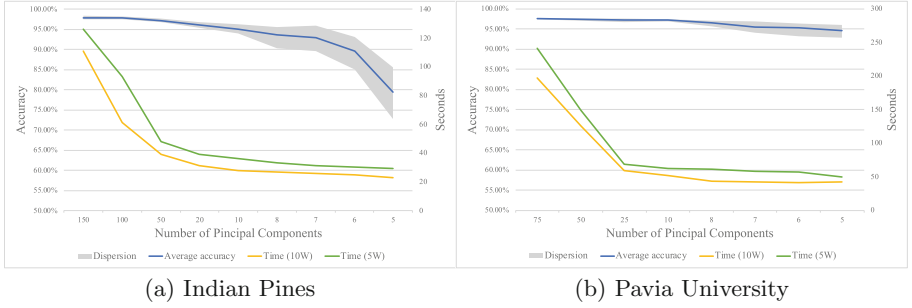
## 4 Results

Firstly, we present in Table 1 the execution times of a few 3D-CNN models from the literature (He et al. [16], Li et al. [21], Hamida et al. [7]), implemented in DeepHyperX and executed on Jetson Nano. We did not include Lee et al. [18], Luo et al., [22] and Chen et al. [9] because they are not competitive in inference execution times (more than 5 min in 10 W modality).

**Table 1.** Execution times on Jetson Nano and classification accuracy of some models from DeepHyperX on IP and PU datasets

Model	Indian Pines			Pavia University		
	Inference Time		Accuracy	Inference Time		Accuracy
	10 W	5 W		10 W	5 W	
He et al.	01:05	01:24	95.35%	04:19	06:10	96.14%
Li et al.	00:23	00:27	97.08%	00:56	01:20	97.72%
Hamida et al.	00:24	00:29	85.72%	01:07	01:40	97.59%

The best results in terms of accuracy and execution time for both datasets and power modalities are those of Li et al. All the subsequent reasonings and tests will suppose the adoption of this promising model, considering our context of on-board processing of remote sensing data.



**Fig. 4.** Accuracy and execution time of inference using our pipeline with several numbers  $K$  of Principal Components.

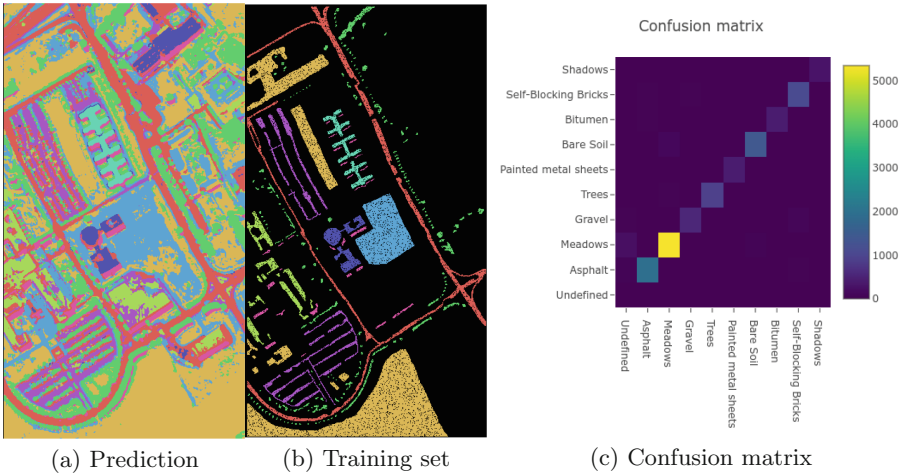
By applying the PCA, if we decrease the number of Principal Components  $K$  used for the 3D-CNN, classification accuracy and execution time decrease simultaneously (Fig. 4). The curve steepness of the execution time is greater than that of the accuracy. Hence, we do not need to sacrifice significant accuracy to reduce the execution time.

Moreover, the dispersion area of multiple testing increases when using fewer Principal Components. This result means that excessive reduction of the input components during the CNN training implies a less reliable prediction. Indeed, prediction accuracy strictly depends on the model and the training set. We used a random approach to sample the training set when repeating the tests, so we trained with different random samples each time. Consequently, an increasing accuracy dispersion means that the training samples' choice becomes highly relevant. Hence, the excessive reduction in the number of principal components directly impacts the training quality of the neural network.

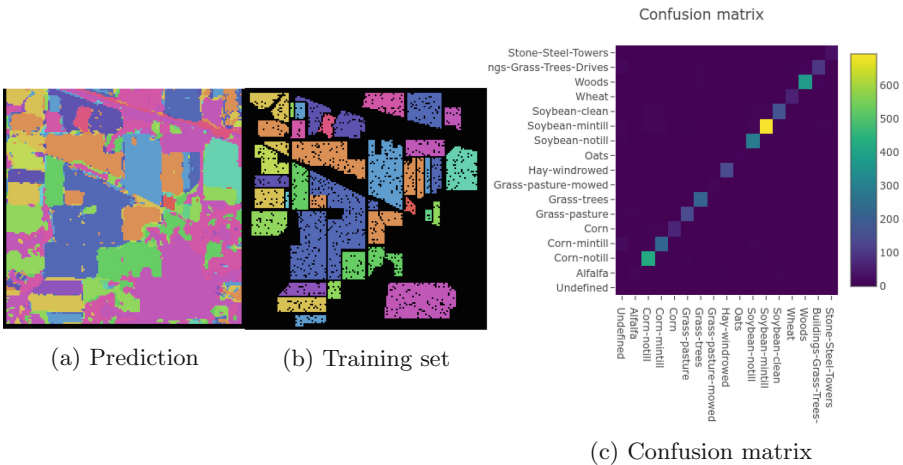
Following these observations, we think a good trade-off between accuracy and execution time is  $K = 50$  for IP (Fig. 4a), thus reducing the dimension of the initial dataset by 75%. However, if we need to reduce the execution time further, we could choose  $K = 10$  while keeping a 95% prediction accuracy and a 95% dimensional reduction. The numbers change for PU (Fig. 4b) since the hyperspectral bands are fewer. To maintain at least a 97% accuracy, we can choose an optimal  $K = 10$ , obtaining an approximate 90% reduction in the dataset dimension. On the other hand, we can choose  $K = 5$  for an approximate 95% dimensional reduction and a 95% accuracy. We will use the  $K$  values mentioned above to control the prediction accuracy in the following testing.

Visual comparisons between ground truth and prediction for PU (Fig. 5) and IP (Fig. 6) datasets show that the results of our pipeline represent reliable classifications, as confirmed by the diagonal of the confusion matrices in Fig. 5c and Fig. 6c.

Regarding power consumption, we tested our pipeline on the Jetson Nano using both energy modalities: 10 W (Table 2) and 5 W (Table 3). We can notice that the advantage of using our pipeline with PCA preprocessing for the PU



**Fig. 5.** Pavia University prediction with 95% accuracy (a), the training set with 70% samples from ground truth (class Undefined in black) (b), and relative confusion matrix (c).



**Fig. 6.** Indian Pines prediction with 95% accuracy (a), the training set with 70% samples from ground truth (class Undefined in black) (b), and relative confusion matrix (c).

dataset is evident, as it halves energy consumption and markedly reduces the execution time. There is a slight improvement for the IP dataset when using the 10 *W* modality. Instead, with 5 *W*, we only see an improvement in energy consumption when reducing the accuracy to 95%. This limitation is due to the HSI shape, which is spatially small but spectrally big in IP, and therefore the GPU-parallel PCA weights more on total time and energy consumption. To bet-



**Table 2.** Comparison of energy consumption and execution times on Jetson Nano (10 W modality) for Li et al. model without and with PCA preprocessing

	Without PCA		With PCA accuracy 95%		With PCA accuracy 97%	
	Secs	Joules	Secs	Joules	Secs	Joules
Pavia University	80	320.3	42	152.41	52	199.7
IndianPines	39	175.78	28	90.80	39	173.25

**Table 3.** Comparison of energy consumption and execution times on Jetson Nano (5 W modality) for Li et al. model without and with PCA preprocessing

	Without PCA		With PCA accuracy 95%		With PCA accuracy 97%	
	Secs	Joules	Secs	Joules	Secs	Joules
Pavia University	107	354.97	48	166.21	62	204.07
IndianPines	50	148.58	36	102.3	48	186.24

ter understand the energetic performance of our proposed pipeline, we compare in Table 4 several items. We calculated energy consumption on the RTX 2060 in joules, multiplying the GPU’s Thermal Design Point (160 W) by the execution time.

It appears evident that for the overall measuring, the execution on the RTX is less time-consuming at the cost of more energy absorption. On the other hand, using both Jetson’s modalities for the PCA implemented with cuBLAS, we measured a saving of about 95% of energy compared to RTX, but with an increment of only 55 – 61% in the execution time. This result is fascinating when considering possible future data processing implementations at the Edge. Regarding the performance of the PCA from scikit-learn, it does not exploit the GPU and therefore is non-competitive.

Looking at overall measuring, including 3D-CNN, we report an increase of about 90% in execution time, saving 70 – 80% in terms of power consumption. That proportion is not promising as the PCA case, probably due to PyTorch inefficiencies. However, it is still an interesting option when connection bandwidth is critical. If we think of a situation with a poor transfer connection, processing at the Edge can reduce bandwidth requests. For example, in our case of HSI classification, the PU dataset consists of 33.2 MB, while the classification output is an image of 610x340 bytes.

**Table 4.** Comparison of energy consumption and execution times on both platforms, isolating PCA preprocessing contributions. In italic, measures for PCA using scikit-learn as reference. The totals refer to PCA with cuBLAS plus inference, setting the accuracy to 95%.

			RTX (160 W)	Jetson (10 W)	Jetson (5 W)
PaviaUniversity	<b>PCA Cublas</b>	Joules	28.8	1.34	1.21
		Secs	0.18	0.4	0.42
	<i>PCA scikit-learn</i>	<i>Joules</i>		<i>18.02</i>	<i>23.11</i>
		<i>Secs</i>	<i>3.09</i>	<i>5.38</i>	<i>7.64</i>
	<b>Total</b>	Joules	481.6	152.41	138.59
		Secs	3.01	42	48
IndianPines	<b>PCA Cublas</b>	Joules	43.2	2.49	2.10
		Secs	0.27	0.7	0.7
	<i>PCA scikit-learn</i>	<i>Joules</i>		<i>12.12</i>	<i>17.10</i>
		<i>Secs</i>	<i>1.48</i>	<i>3.41</i>	<i>5.7</i>
	<b>Total</b>	Joules	432.0	90.80	102.3
		Secs	0.27	28	36

## 5 Conclusions

This work shows an innovative perspective on the HSI classification problem contextualized in High-Performance Edge Computing. By adopting the Nvidia Jetson Nano system-on-chip, which can be attached to remote sensors of various types, we developed an HSI classifier optimized for the Edge to enable onboard processing. In such a context, the processing time is focal; therefore, we chose the most promising 3D-CNN model in prediction accuracy and inference time using a GPU.

Then, to further speed up the processing, we applied a Principal Component Analysis to the original dataset to obtain up to a 90% reduction in size without significantly depleting accuracy.

To exploit the acceleration available on the Jetson Nano and achieve high performance, we implemented a GPU-parallel version of the PCA in CUDA. Furthermore, we analyzed the energy absorption on the Jetson Nano to identify the best energy configuration for our problem. The 10W modality resulted in the shortest execution time, even if it did not correspond to greater energy consumption for both considered datasets.

Results are encouraging to further investigate the problem by analyzing datasets from more recent sensors that are not yet publicly available. Moreover, additional analysis of the two energy modalities in the Jetson Nano on other applications can result in possibly interesting evidence about Edge energy consumption. Another improvement could be considering a scenario where the GPU is remoted and the actual computation executed on low-power devices or single-board computers, as in [24, 26].

## References

1. Ai, Y., Peng, M., Zhang, K.: Edge cloud computing technologies for internet of things: a primer. *Digit. Commun. Netw.* **4**, 77–86 (2017)
2. Andrecut, M.: Parallel GPU implementation of iterative PCA algorithms. *J. Comput. Biol.* **16**(11), 1593–1599 (2009)
3. Armbrust, M., et al.: A view of cloud computing. *Commun. ACM* **53**(4), 50–58 (2010)
4. Audebert, N.: Deephyperx. <https://github.com/nshaud/DeepHyperX>
5. Audebert, N., Le Saux, B., Lefèvre, S.: Deep learning for classification of hyperspectral data: a comparative review. *IEEE Geosci. Remote Sens. Mag.* **7**(2), 159–173 (2019)
6. Audebert, N., Le Saux, B., Lefèvre, S.: Semantic segmentation of earth observation data using multimodal and multi-scale deep networks. In: Lai, S.-H., Lepetit, V., Nishino, K., Sato, Y. (eds.) *ACCV 2016. LNCS*, vol. 10111, pp. 180–196. Springer, Cham (2017). [https://doi.org/10.1007/978-3-319-54181-5\\_12](https://doi.org/10.1007/978-3-319-54181-5_12)
7. Ben Hamida, A., Benoit, A., Lambert, P., Ben Amar, C.: 3-D deep learning approach for remote sensing image classification. *IEEE Trans. Geosci. Remote Sens.* **56**(8), 4420–4434 (2018)
8. Chang, C.I.: *Hyperspectral Imaging: Techniques for Spectral Detection and Classification*, vol. 1. Springer, Cham (2003)
9. Chen, Y., Jiang, H., Li, C., Jia, X., Ghamisi, P.: Deep feature extraction and classification of hyperspectral images based on convolutional neural networks. *IEEE Trans. Geosci. Remote Sens.* **54**(10), 6232–6251 (2016)
10. De Lucia, G., Lapegna, M., Romano, D.: Towards explainable AI for hyperspectral image classification in edge computing environments. *Comput. Electr. Eng.* **103**, 108381 (2022)
11. Ding, N., Wagner, D., Chen, X., Pathak, A., Hu, Y.C., Rice, A.: Characterizing and modeling the impact of wireless signal strength on smartphone battery drain. *ACM SIGMETRICS Perform. Eval. Rev.* **41**(1), 29–40 (2013)
12. Grahn, H., Geladi, P.: *Techniques and Applications of Hyperspectral Image Analysis*. John Wiley, Hoboken (2007)
13. Greenberg, A., Hamilton, J., Maltz, D.A., Patel, P.: The cost of a cloud: research problems in data center networks. *SIGCOMM Comput. Commun. Rev.* **39**(1), 68–73 (2009)
14. Grupo de Inteligencia Computacional (GIC): Hyperspectral dataset. [http://www.ehu.eus/ccwintco/index.php/Hyperspectral\\_Remote\\_Sensing\\_Scenes](http://www.ehu.eus/ccwintco/index.php/Hyperspectral_Remote_Sensing_Scenes)
15. Ha, K., Chen, Z., Hu, W., Richter, W., Pillai, P., Satyanarayanan, M.: Towards wearable cognitive assistance. In: *Proceedings of the 12th Annual International Conference on Mobile Systems, Applications, and Services, MobiSys 2014*, pp. 68–81. Association for Computing Machinery, New York (2014)
16. He, M., Li, B., Chen, H.: Multi-scale 3D deep convolutional neural network for hyperspectral image classification. In: *2017 IEEE International Conference on Image Processing (ICIP)*, pp. 3904–3908 (2017)
17. Lapegna, M., Balzano, W., Meyer, N., Romano, D.: Clustering algorithms on low-power and high-performance devices for edge computing environments. *Sensors* **21**(16), 5395 (2021)
18. Lee, H., Kwon, H.: Going deeper with contextual CNN for hyperspectral image classification. *IEEE Trans. Image Process.* **26**(10), 4843–4855 (2017)

19. Li, J., Cui, R., Li, B., Li, Y., Mei, S., Du, Q.: Dual 1d–2d spatial-spectral CNN for hyperspectral image super-resolution. In: IGARSS 2019–2019 IEEE International Geoscience and Remote Sensing Symposium, pp. 3113–3116 (2019)
20. Li, Y., Zhang, H., Shen, Q.: Spectral-spatial classification of hyperspectral imagery with 3D convolutional neural network. *Remote Sens.* **9**(1), 67 (2017)
21. Li, Y., Zhang, H., Shen, Q.: Spectral-spatial classification of hyperspectral imagery with 3D convolutional neural network. *Remote Sens.* **9**(1) (2017)
22. Luo, Y., Zou, J., Yao, C., Zhao, X., Li, T., Bai, G.: HSI-CNN: a novel convolution neural network for hyperspectral image. In: 2018 International Conference on Audio, Language and Image Processing (ICALIP), pp. 464–469 (2018)
23. Makantasis, K., Karantzalos, K., Doulamis, A., Doulamis, N.: Deep supervised learning for hyperspectral data classification through convolutional neural networks. In: 2015 IEEE International Geoscience and Remote Sensing Symposium (IGARSS), pp. 4959–4962. IEEE (2015)
24. Marcellino, L., et al.: Using GPGPU accelerated interpolation algorithms for marine bathymetry processing with on-premises and cloud based computational resources. In: Wyrzykowski, R., Dongarra, J., Deelman, E., Karczewski, K. (eds.) PPAM 2017. LNCS, vol. 10778, pp. 14–24. Springer, Cham (2018). [https://doi.org/10.1007/978-3-319-78054-2\\_2](https://doi.org/10.1007/978-3-319-78054-2_2)
25. Marmanis, D., Wegner, J.D., Galliani, S., Schindler, K., Datcu, M., Stilla, U.: Semantic segmentation of aerial images with an ensemble of CNSS. *ISPRS Ann. Photogrammetry Remote Sens. Spat. Inf. Sci.* **2016**(3), 473–480 (2016)
26. Montella, R., Giunta, G., Laccetti, G.: Virtualizing high-end GPGPUS on arm clusters for the next generation of high performance cloud computing. *Cluster Comput.* **17**(1), 139–152 (2014)
27. Raychaudhuri, D., Nagaraja, K., Venkataramani, A.: Mobilityfirst: a robust and trustworthy mobility-centric architecture for the future internet. *ACM SIGMOBILE Mob. Comput. Commun. Rev.* **16**(3), 2–13 (2012)
28. Rodarmel, C., Shan, J.: Principal component analysis for hyperspectral image classification. *Surv. Land Inf. Syst.* **62**(2), 115–123 (2002)
29. Romano, D., Lapegna, M.: A GPU-parallel image coregistration algorithm for InSar processing at the edge. *Sensors* **21**(17), 5916 (2021)
30. Sabins, F.F.: Remote sensing for mineral exploration. *Ore Geol. Rev.* **14**(3–4), 157–183 (1999)
31. Slavkovikj, V., Verstockt, S., De Neve, W., Van Hoecke, S., Van de Walle, R.: Hyperspectral image classification with convolutional neural networks. In: Proceedings of the 23rd ACM International Conference on Multimedia, pp. 1159–1162 (2015)
32. Tharwat, A.: Principal component analysis-a tutorial. *Int. J. Appl. Pattern Recognit.* **3**(3), 197–240 (2016)
33. Volpi, M., Tuia, D.: Dense semantic labeling of subdecimeter resolution images with convolutional neural networks. *IEEE Trans. Geosci. Remote Sens.* **55**(2), 881–893 (2016)