Efficient Semantic Segmentation with Hyperspectral Images

Fernando Peña, Pilar Vidal Aguilar, Darío Suárez Gracia, and Ana C. Murillo

DIIS, I3A, Universidad de Zaragoza, Spain

Abstract. Many robotic application domains, such as agriculture, environmental management, or industrial automation, could benefit from hyperspectral images and their ample spectrum information. However, the huge volume of information makes automated semantic segmentation of hyperspectral images both a daunting task and a barrier to its adoption. This paper explores alternatives to obtain accurate and more efficient results for semantic segmentation in hyperspectral imagery. The proposed pipeline combines standard methods to reduce the dimensionality of the input, followed by an efficient deep neural network to segment the information. Our results show that this pipeline performs better than available related work solutions in terms of accuracy and performance. Besides, this work makes a detailed study of the trade-offs between the amount of information received by the segmentation network and the accuracy, execution time, and energy efficiency of the solution.

Keywords: hyperspectral, segmentation, images, neural networks

1 Introduction

Hyperspectral Imaging (HSI) is a technique that captures a wide spectrum of light at different wavelengths, instead of just the primary colors as conventional cameras do. The images captured by HSI sensors can not only be used to characterize objects based on their shape and color, but also allow the identification of the materials. This brings great potential for automated detection and classification tasks and makes HSI very useful for a broad range of robotic applications, such as monitoring in security, agriculture, or environmental tasks [3, 5], in automated industrial tasks such as recycling [6], or in assistive medical applications [11].

Implementing novel solutions for all these applications requires automatically segmenting the content of the hyperspectral data. Semantic segmentation aims to assign a label to each pixel of an image These individual pixels will shape the blobs belonging to each category. Each blob belongs to a particular class, and this allows differentiation between categories. As in many other fields, deep learning techniques are pushing the state of the art on segmentation for hyperspectral data. However, processing hyperspectral data is particularly complex in terms of computational requirements, and there is a lack of large annotated datasets, which hinders the training of deep neural networks. In addition, most hyperspectral scene datasets are composed by a single very large image capture, corresponding

to a limited location[1]. This makes supervised training challenging, especially for deep learning models, which typically require large training sets. The solution is splitting part of the image for training and another for testing. However, the splitting has to be done very carefully and many existing approaches often lack details about this crucial procedure. For example, suppose just split the image into a grid of patches, and randomly build the train and test sets. In that case, we may end up with neighboring patches in training and testing which are almost identical, provoking overfitting and giving a false sense of good performance of the models.

This work aims to analyze these challenges and pave the way towards novel solutions for efficient and accurate semantic segmentation. The proposed work introduces a model that is able to process and predict a whole image at once, presenting it as a new method compared to commonly used ones that process small patches. In particular, the contributions of this work are the following:

- A proposal for an evaluation framework, based on existing benchmarks, for a more challenging and realistic evaluation of HSI, despite the small size of existing labeled datasets.
- A study of a new pipeline for hyperspectral image segmentation based on an efficient segmentation network, rather than frequently used image classification networks used in related works, with promising results improving available solutions.
- A performance analysis of our pipeline to detect the best trade-offs between accuracy and performance, and the possible bottlenecks.

2 Related Work

This section is organized in two blocks, summarizing the following topics: deep neural networks used with hyperspectral images for patchwise classification tasks and existing general-purpose segmentation networks.

Patchwise classification solutions. The flexibility of deep learning methods for modeling complex data is very appealing in the case of hyperspectral imaging, where every single image contains a large amount of information. One of the mainstream deep learning architectures used for HSI data are convolutional neural networks (CNNs), since they efficiently integrate spectral features along with spatial information from the HSI data.

In most of the literature regarding HSI segmentation, CNNs are used to classify small patches from the typically large hyperspectral images. We name this group of approaches *patchwise* classification solutions. These methods split the complete HSI image into small patches (around 25×25 pixels), and aim to predict a class label for each patch. Later, all the predicted patches are joined together to form the final segmented image. They do not work towards a more complex scene understanding by analyzing all the context of the image, but only

[1] https://www.ehu.eus/ccwintco/index.php/Hyperspectral_Remote_Sensing_Scenes

predict the label for each individual patch. Two of the works that follow this strategy and have shown top results in public benchmarks are SpectralNET [4] and HybridSN [9].

A key aspect regardless of the architecture used for the classification, is the preprocessing applied to the images before they are processed by the segmentation network. Most HSI pipelines perform a linear transformation of the bands to reduce the dimensionality of the network input data and make the training and inference process more manageable. Specifically, SpectralNET performs factor analysis (FA), which tries to obtain a reduced number of variables that can still represent well most of the information. The authors use representation reduced to only three components. HybridSN uses principal component analysis (PCA), which reprojects the data into a different space so that the first components account for the maximum variation in the data. In most cases, models end up using between 15 and 30 dimensions.

Semantic segmentation solutions. Semantic segmentation is a broadly studied problem in computer vision that attempts to classify all the pixels in the image, commonly using encoder-decoder architectures, such as SegNet [2]. However, these techniques are not commonly used yet for HSI segmentation, although they present significant advantages and promising results [7]. Segmentation architectures are able to extract features from the context of each pixel. They do not focus exclusively on a single patch around the pixel (as the patchwise classification does) but on surrounding regions at different scales. These models do not include final fully-connected layers so the internal state of the network always maintains the spatial meaning. The result is that the edges and shapes are better defined and more coherent.

Besides, from an efficiency perspective, semantic segmentation models have the advantage that they are able to classify all the pixels of the image in one go, while when performing inference with patchwise methods, we need to run the inference network for each of the patches.

Many recent works propose how to improve the efficiency of the solutions, to enable real-world robotic applications [8, 1], such as autonomous driving or different monitoring tasks. We study the application of this type of efficient architectures to hyperspectral image segmentation.

3 Method

The pipeline proposed for semantic segmentation consists of two steps as shown in Fig. 1. After receiving a batch of images, we first perform dimensionality reduction, and, second, we perform semantic segmentation using a modified version of MiniNet-v2 [1] that accepts images with any number of channels.

Reduction Hyperspectral data is highly dimensional and can have from tens to hundreds of bands, so a common preprocessing step is to reduce them to facilitate training and inference. This reduction can be obtained in different ways. In this

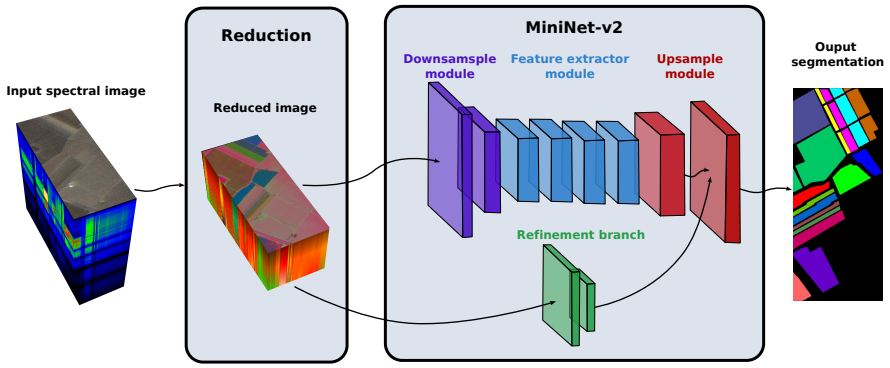


Fig. 1: Proposed semantic segmentation pipeline. The original hyperspectral image is reduced to a lower number of dimensions and then processed by a segmentation network [1] to obtain the final segmentation result.

work, we study the two most common approaches in the literature: factor analysis and principal component analysis [10]. We reduce the data to 3 or 20 components, to facilitate comparisons with available previous work architectures [4, 9].

To consider a realistic setup, dimensionality reduction models need to be only learned with the training sample split. When testing, the images are projected in the space computed with training samples. That way, the evaluation metrics are more realistic and representative of a pipeline that continuously receives new images unseen at the time of model learning.

Segmentation MiniNet-v2 [1] is a CNN for semantic segmentation that is efficient in terms of memory, speed, and computation, and can help in environments where computational and temporal restrictions are tight. It uses convolutional layers and its architecture follows an encoder-decoder structure, inspired by the ERFNet [8] model, trying to obtain the highest efficiency. It is able to adapt to the availability of GPU to obtain the best results and can also be adapted to run on CPU.

MiniNet-v2 has four main blocks. The first is the *Downsample* block, which combines downsampling operations with convolutional layers. The second one is the *Feature extractor*, which is in charge of encoding the information using consecutive convolutional modules. The third block is the *Refinement* block, which extracts spatial and high resolution features from the input image. The last one is the decoder or *Upsample* block, that upsamples the outputs of the *Feature extractor* and the *Refinement* blocks added together.

The original implementation of MiniNet-v2 only accepted RGB images. In order to support inputs with an arbitrary number of channels, the data loader was adapted. Since the network follows an encoder-decoder architecture, it can scale with the number of channels, so it was not necessary to modify its internal structure.

4 Evaluation Methodology

4.1 Metrics

We have divided the evaluation into two categories. We evaluate the accuracy of the segmentation results but also perform a detailed efficiency analysis to detect bottlenecks in the pipeline.

Segmentation Evaluation. In order to compare the results between different configurations of the pipeline and with respect to existing methods we use the following metrics:

- **Overall Accuracy (OA).** The amount of correctly classified test samples divided by the number of samples (also known as test accuracy).
- **Average Accuracy (AA).** This metric represents the average of the class-wise classification accuracy:

$$\mathrm{AA} = \frac{\sum_{c \in C} \mathrm{A}_c}{|C|}, \tag{1}$$

  where C denotes the set of the different classes, and A_c is the classification accuracy for class c.
- **Mean intersection over union (MIoU).** This is the most relevant metric in semantic segmentation literature. The intersection over union per class, IoU_c, is the area of overlap (intersection) between the predicted segmentation and the ground truth, divided by the area of their union. The MIoU is obtained by averaging the IoU of each class:

$$\mathrm{MIoU} = \frac{\sum_{c \in C} \mathrm{IoU}_c}{|C|}, \tag{2}$$

  where IoU_c is the IoU for class c.

Performance Evaluation. The efficiency of each method is measured in terms of inference time per patch and per pixel. In addition, we perform a more comprehensive evaluation of the proposed pipeline, measuring the time needed for dimensionality reduction and inference separately and the energy consumption of the CPU and GPU.

4.2 Proposed Benchmark

We decided to choose the Salinas dataset[2] to evaluate the pipeline since its classes are well distributed across the image. The dataset was taken at a test site in Salinas Valley (California) using an AVIRIS Sensor. The image gathered by the sensor has a dimension of $512\,\mathrm{px} \times 217\,\mathrm{px} \times 224\,\mathrm{bands}$ and the spatial resolution

[2] https://paperswithcode.com/dataset/salinas

is 3.7 meters per pixel. This dataset contains sixteen types of land cover including different types of crops (see Fig. 2). However, many classes are equivalent, only differing in the state of growth of the crops, so we fused them to form a new version of the dataset containing seven different classes.

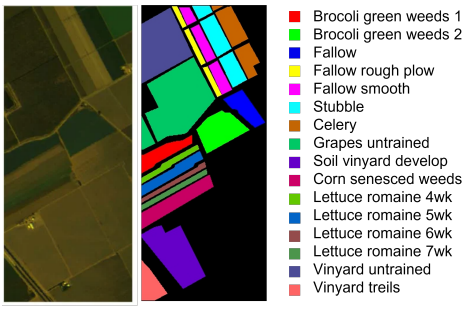


Fig. 2: Salinas dataset. On the left, the false color map, and on the right, the classified map. Black pixels are labeled as *None*.

The dataset was then carefully split into non-overlapping training and testing sets. First, we split the image into four quarters. We discarded the bottom-right one because most pixels were unlabeled (labeled as *None* or zero). Each of the remaining quarters were subsequently divided into three parts and finally distributed into training and testing sets as shown in Fig. 3.

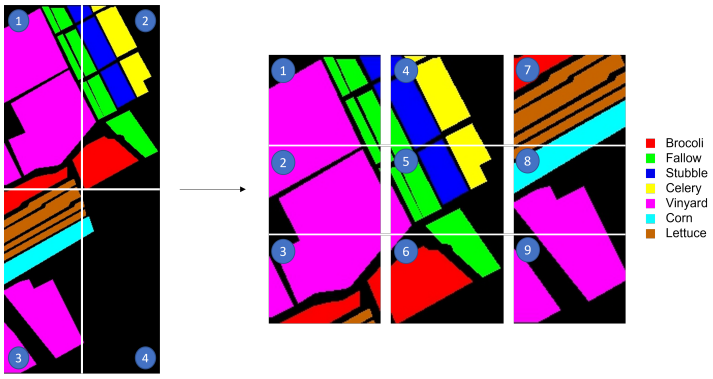


Fig. 3: Salinas dataset relabeled and split into regions to be distributed into training and testing sets. Quarter number 4 is discarded due to emptiness. The other three quarters are split into three regions each. Regions 1,5, 6, 7, and 9 will be for training, and regions 2, 3, 4, and 8 for testing.

5 Experimental Evaluation

5.1 Experimental Setup

The accuracy and efficiency experiments were carried out on an Intel Core i7-6700HQ processor and an NVIDIA GeForce GTX 970M GPU running Ubuntu 20.04.2 LTS and CUDA 11.6. We used the original MiniNet-v2 implementation in Tensorflow and Scikit-learn's implementation of FA and PCA. Energy measurements were carried out with Intel Running Average Power Limit and NVIDIA System Management Interface.

There are four main configurations of the pipeline under test, which are related to the specific configuration of the dimensionality reduction step: FA 3, FA 20, PCA 3, and PCA 20, which correspond to factor analysis and principal component analysis with 3 and 20 components, respectively. These tests will show the impact that dimensionality reduction has in terms of accuracy and efficiency.

For measuring the accuracy and MIoU, we gathered the regions of the Salinas dataset chosen for testing on a single $340 \times 108 \times 204$ image and, after reducing it, classified it with a model trained during 250 epochs.

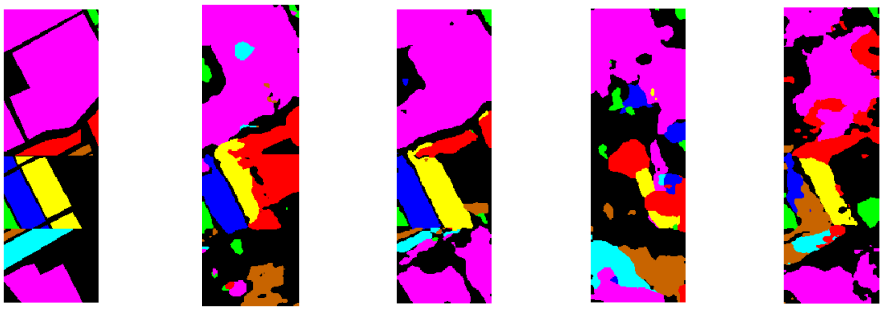
For the efficiency tests, we were interested in assessing the scalability of the system, so we measured the time and energy consumption when performing inference on batches of different sizes, composed by that $340 \times 108 \times 204$ image repeated several times. To ensure the accuracy of the results, measurements were repeated 20 times and averaged.

5.2 Accuracy

Table 1 shows the accuracy and MIoU for each configuration. It is clear that factor analysis is a more suitable method for dimensionality reduction in this case. Figure 4 shows the segmentation results, which confirm, qualitatively, that the results obtained with FA are the best. It is also important to note that using more dimensions consistently improves the metrics, so it would be beneficial to use the maximum number of components possible within the established efficiency limits.

Table 1: Segmentation quality metrics on Salinas dataset using different configurations of our pipeline.

Model	Reduction method	Comps.	Test OA%	Test AA%	Test MIoU%
MiniNet-v2	FA	3	66.10	56.17	38.21
MiniNet-v2	FA	20	**81.13**	**72.66**	**59.63**
MiniNet-v2	PCA	3	43.51	27.88	16.99
MiniNet-v2	PCA	20	67.42	61.19	39.71



(a) GT (b) FA 3 (c) FA 20 (d) PCA 3 (e) PCA 20

Fig. 4: Segmentation results on Salinas datasets using different configurations of our pipeline.

5.3 Efficiency

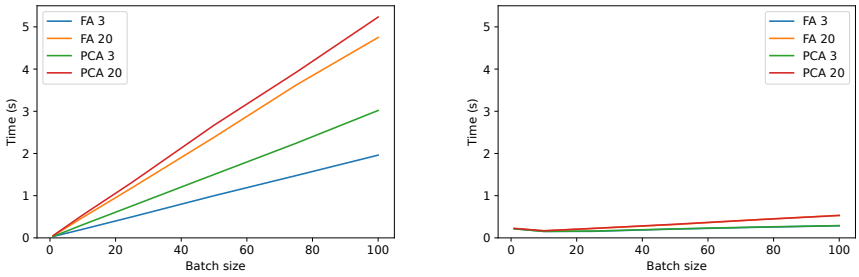
This sub-section compares the efficiency of the different reduction methods in terms of execution time, throughput, and energy.

Figure 5 shows the total execution time for the reduction and inference stages on the left and right sides, respectively, for batch sizes between 1 and 100 images of size $340 \times 108 \times 204$. The reduction stage runs on the CPU, while the GPU executes the inference. The extra compute capabilities of the GPU coupled with the highly optimized neural network code make the reduction time the main bottleneck regardless of the batch size. In fact, the inference delay remains almost constant below 540 ms. On the contrary, the number of output components drives the execution time, and the faster configuration, FA 3, is around $5\times$ faster than the slowest configuration, PCA 20.

The reduced execution time of FA 3 helps this configuration to reach the highest throughput as shown in Fig. 6(a). Interestingly, FA 3 is the only configuration that could be used with a camera delivering 30 frames per second because PCA's highest throughput stays below 15 FPS.

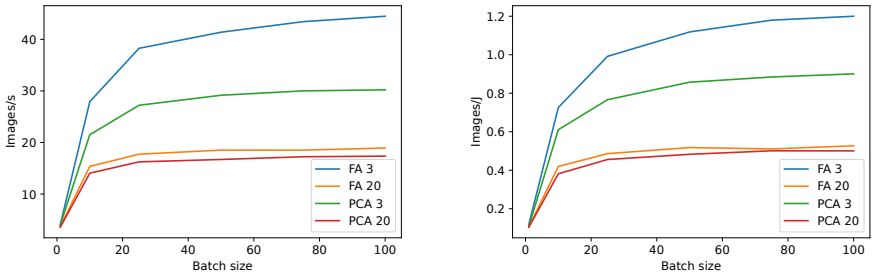
Figure 6(b) shows the energy efficiency measured in energy per joules, or throughput divided by power. In other words, how many images can be semantically segmented per unit of consumed energy. The correlation between energy efficiency and throughput indicates that average power is almost the same regardless of the reduction algorithm. This is true because, as Fig. 7b shows, the GPU consumption dominates the overall energy consumption.

Interestingly, the analysis of throughput and energy efficiency indicates that depending on the environment; e.g., a battery-powered robot, the optimization focus should be on reducing the GPU energy, which could also negatively impact execution time. However, in "plugged" always-on systems, such as processing garbage to be recycled, the focus should be on reducing the execution time on the CPU, by either optimizing the CPU execution or moving the computation to the GPU in case the segmentation does not utilize all resources.



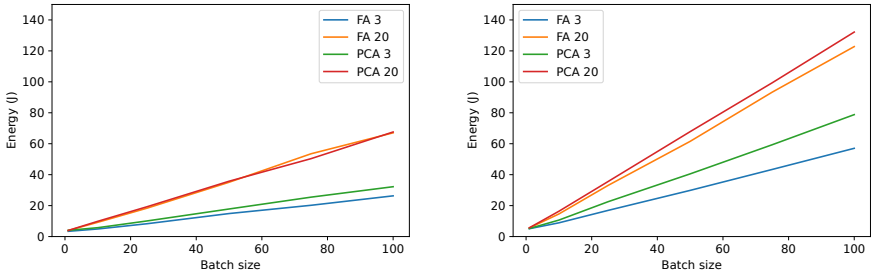
(a) Reduction (b) Inference

Fig. 5: Split total execution time for a single batch composed by multiple images for reduction and inference steps.



(a) Throughput (b) Energy

Fig. 6: Efficiency analysis: (a) Throughput for all configuration under test. (b) Overall energy efficiency.



(a) CPU (b) GPU

Fig. 7: Energy efficiency per device.

Table 2: Salinas dataset results using different configurations and models. The inference time per pixel is the time per patch divided by the number of pixels in a patch. All results were obtained including *None* patches in the training/testing sets.

Model	Patch size	Comps.	Test OA%	Test AA%	Test MIoU%	Inference time /patch (ms)	Inference time /pixel (ms)
SpectralNET	24×24	3	76.7	71.2	54.5	42	0.073
HybridSN	25×25	15	80.5	75.1	59.8	35	0.056
MiniNet-v2 FA	340×108	3	66.1	56.2	38.2	75	0.002
MiniNet-v2 FA	340×108	20	81.1	72.7	59.6	78	0.002

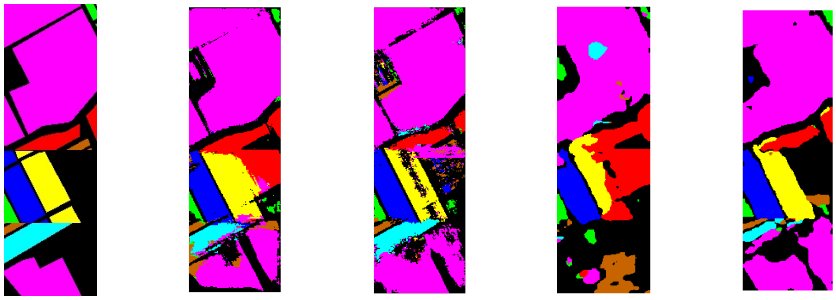
5.4 Comparison with Previous Work

In this section we compare the performance of the proposed pipeline with previously discussed patchwise methods: SpectralNET[3] and HybridSN[4]. We selected these methods since they had available repositories and published top results on public benchmarks. For these methods, we used the configuration suggested in the original works regarding patch size and the number of components used after dimensionality reduction. In this experiment, the dimensionality reduction models were trained with all the data available (train and test) for all models, to have a common setup for comparison.

Although some of the published results do, we did not ignore *None* (unlabeled) patches prior to learning, since doing it is not realistic in real-world scenarios, where we do not have access to the ground truth labels. So, for a fair comparison, we re-trained all patchwise models without removing *None* patches. They were trained with the training/testing division explained in Sect. 4.2. We used the recommended training hyperparameters for both patchwise models in their respective implementations. All configurations of MiniNet-v2 were trained during 250 epochs with a learning rate of 0.0002 and batch size 8.

Table 2 shows the accuracy and efficiency results, while Fig. 8 shows the resulting segmentations. MiniNet-v2 with factor analysis and 20 components obtains similar results to the top-performing patchwise model (HybridSN) while being much more efficient. Note that even though MiniNet-v2 uses more components and processes bigger patches, when we normalize the inference time per pixel after processing all test data, it is significantly lower than the patchwise alternatives. In addition, MiniNet-v2's outputs are already segmented images. Completely segmenting an image with a patchwise model requires running the inference network for a patch around each of its pixels. Note that if done sequentially, classifying a 340×108 image with HybridSN takes at least 2 s, if we use the minimum number of patches of 25×25 to cover the whole image, and much

[3] https://github.com/tanmay-ty/SpectralNET
[4] https://github.com/gokriznastic/HybridSN



(a) GT (b) SpectralNET (c) HybridSN (d) MN-v2 FA 3 (e) MN-v2 FA 20

Fig. 8: Visualization of the ground truth segmentation (a) and predicted scenes with the patchwise models (b, c) and MiniNet-v2 (d, e). Each color represents a different semantic class.

more with the suggested setup of evaluating a patch around each image pixel (up to several minutes).

6 Conclusions and Future Work

This work presents a pipeline for semantic segmentation of hyperspectral images focused on efficiency, which is essential to enable the use of this kind of sensors in many robotics applications. The two-stage pipeline is composed of a reduction, which identifies the primary source of information from the images, followed by a segmentation, which labels the pixels with the semantic category. This approach reduces the execution time with similar or better results compared to available solutions. To ensure a fair comparison, our work proposes a new benchmarking strategy for existing hyperspectral datasets, which are of limited size and challenging to perform realistic evaluations.

Besides, the paper includes an analysis of the performance and energy efficiency to guide optimization depending on the application domain. For example, it highlights where to reduce energy in battery-powered platforms or increase performance for non-energy-constrained robots.

Acknowledgments

This work has been partially supported by grants MIG-20201006 (SEPARA), PID2019-105660RB-C21 and PID2021-125514NB-I00 funded by MCIN/AEI/ 10.13039/501100011033 FEDER, UE, and Government of Aragon projects FSE T45_20R and T58_20R.

References

1. Iñigo Alonso, Luis Riazuelo, and Ana C. Murillo. MiniNet: An efficient semantic segmentation ConvNet for real-time robotic applications. *IEEE Transactions on Robotics*, 36(4):1340–1347, 2020.
2. Vijay Badrinarayanan, Alex Kendall, and Roberto Cipolla. SegNet: A deep convolutional encoder-decoder architecture for image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39(12):2481–2495, 2017.
3. Asher Bender, Brett Whelan, and Salah Sukkarieh. A high-resolution, multimodal data set for agricultural robotics: A *Ladybird*'s-eye view of Brassica. *Journal of Field Robotics*, 37(1):73–96, 2020.
4. Tanmay Chakraborty and Utkarsh Trehan. SpectralNET: Exploring spatial-spectral WaveletCNN for hyperspectral image classification. *arXiv preprint arXiv:2104.00341*, 2021.
5. Maxime Ferrera, Aurélien Arnaubec, Klemen Istenič, Nuno Gracias, and Touria Bajjouk. Hyperspectral 3D mapping of underwater environments. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 3703–3712, 2021.
6. Sathish Paulraj Gundupalli, Subrata Hait, and Atul Thakur. A review on automated sorting of source-separated municipal solid waste for recycling. *Waste Management*, 60:56–74, 2017. Special Thematic Issue: Urban Mining and Circular Economy.
7. Bin Pan, Xia Xu, Zhenwei Shi, Ning Zhang, Huanlin Luo, and Xianchao Lan. DSSNet: A simple dilated semantic segmentation network for hyperspectral imagery classification. *IEEE Geoscience and Remote Sensing Letters*, 17(11):1968–1972, 2020.
8. Eduardo Romera, José M. Álvarez, Luis M. Bergasa, and Roberto Arroyo. ERFNet: Efficient residual factorized ConvNet for real-time semantic segmentation. *IEEE Transactions on Intelligent Transportation Systems*, 19(1):263–272, 2018.
9. Swalpa Kumar Roy, Gopal Krishna, Shiv Ram Dubey, and Bidyut B Chaudhuri. HybridSN: Exploring 3-D–2-D CNN feature hierarchy for hyperspectral image classification. *IEEE Geoscience and Remote Sensing Letters*, 17(2):277–281, 2019.
10. Y. Takane. *Constrained Principal Component Analysis and Related Techniques*. Chapman & Hall/CRC Monographs on Statistics and Applied Prob. Taylor & Francis Group, 2020.
11. Stojan Trajanovski, Caifeng Shan, Pim J. C. Weijtmans, Susan G. Brouwer de Koning, and Theo J. M. Ruers. Tongue tumor detection in hyperspectral images using deep learning semantic segmentation. *IEEE Transactions on Biomedical Engineering*, 68(4):1330–1340, 2021.