IEEE Access
Multidisciplinary : Rapid Review : Open Access Journal

Received 26 December 2023, accepted 22 January 2024, date of publication 1 February 2024, date of current version 12 February 2024.

Digital Object Identifier 10.1109/ACCESS.2024.3360936

RESEARCH ARTICLE

FuSC: Fusing Superpixels for Improved Semantic Consistency

IAN MONTEIRO NUNES [1], MATHEUS B. PEREIRA [2], HUGO OLIVEIRA [3,4], AND JEFERSSON ALEX DOS SANTOS [2,5]

[1]Department of Statistics (DPE), Brazilian Institute of Geography and Statistics (IBGE), Rio de Janeiro 20230-240, Brazil
[2]Department of Computer Science (DCC), Universidade Federal de Minas Gerais (UFMG), Belo Horizonte 31270-901, Brazil
[3]Institute of Mathematics and Statistics (IME), University of São Paulo (USP), São Paulo 05508-220, Brazil
[4]Departamento de Informática (DPI), Universidade Federal de Viçosa (UFV), Viçosa 36570-900, Brazil
[5]Department of Computer Science, The University of Sheffield, S1 4DP Sheffield, U.K.

Corresponding author: Ian Monteiro Nunes (ian.nunes@ibge.gov.br)

This work was supported in part by Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (CAPES), in part by Fundação de Amparo à Pesquisa do Estado de Minas Gerais (FAPEMIG), in part by Fundação de Amparo à Pesquisa do Estado de São Paulo (FAPESP) under Grant 2020/06744-5, and in part by the Serrapilheira Institute under Grant R-2011-37776.

ABSTRACT Open-set segmentation has caught the community's attention only in the last few years, and it is a growing and active research area with many challenges ahead. To better identify open-set pixels, we address two known issues by improving data representation and ensuring semantic consistency in open-set predictions. First, we present a method called Open Gaussian Mixture of Models (OpenGMM) that allows for multimodal statistical distributions in known class pixels using a Gaussian Mixture of Models instead of unimodal approaches, like Principal Component Analysis. The second approach improved semantic consistency by applying a post-processing technique that uses superpixels to enforce homogeneous regions to have similar predictions, rectifying erroneously classified pixels within these regions and providing better delineation of object borders. We also developed a novel superpixel method called Fusing Superpixels for Improved Semantic Consistency (FuSC) that produced more homogeneous superpixels and enhanced, even more, the open-set segmentation prediction. We applied the proposed approaches to well-known remote sensing datasets with labeled ground truth for semantic segmentation tasks. The proposed methods improved the highest AUROC quantitative results for the International Society for Photogrammetry and Remote Sensing (ISPRS) Vaihingen and Potsdam datasets. Using FuSC, we achieved novel open-set state-of-the-art results for both datasets, improving AUROC results from 0.850 to 0.880 (3.53%) for Vaihingen and 0.764 to 0.797 (4.32%) for Potsdam datasets. The official implementation is available at: https://github.com/iannunes/FuSC.

INDEX TERMS Convolutional neural network, open-set, segmentation, remote sensing, semantic consistency, superpixel, clustering.

I. INTRODUCTION

Remote sensing acquisition technologies have been in constant development since the 1960s, providing sensors with a myriad of new electromagnetic spectral encoding capabilities and leading to a continuous increase in the volume of daily collected data. The development of these technologies and an increased capacity to produce relevant information for a wide range of applications turned the automatic analysis of images into one of the most actively researched fields within the remote sensing community [1].

Since the Gestalt movement, it is known that image segmentation and clustering play a relevant role in human perception [2]. Many different applications can benefit from semantic segmentation of remote sensing images, such as urban planning/mapping and change detection [3], population estimation [4], real-estate management [5].

The associate editor coordinating the review of this manuscript and approving it for publication was Zhan-Li Sun.

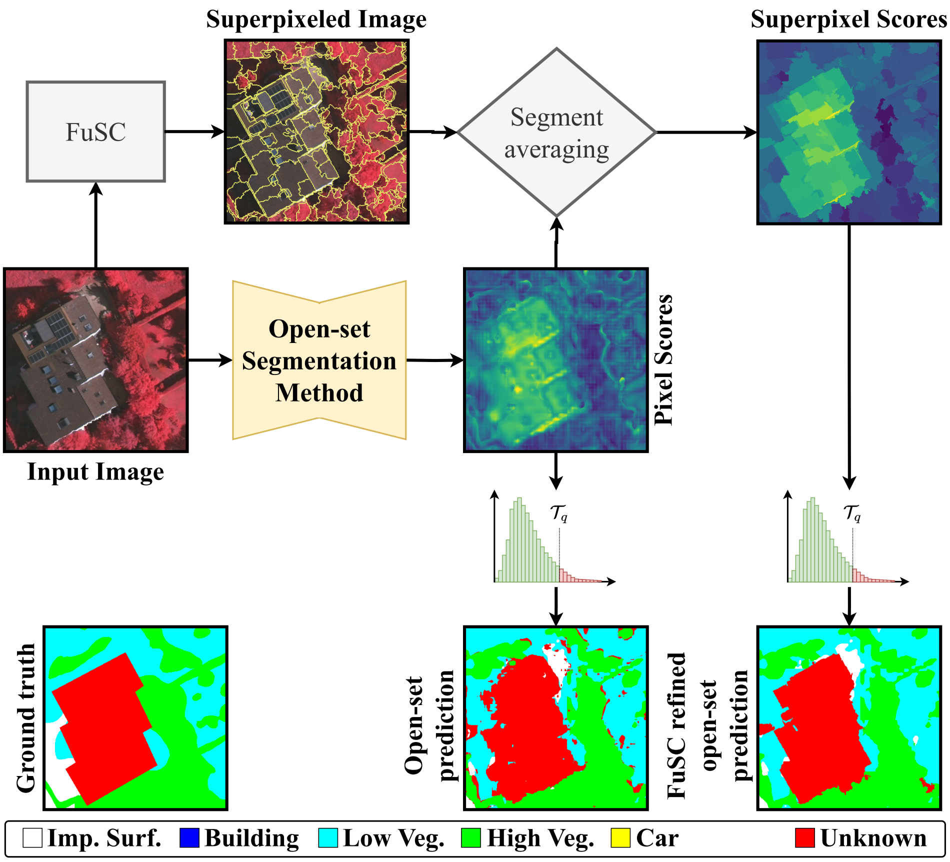


FIGURE 1. Overview of the proposed approach for Open Set Semantic Segmentation (OSS) with a post-processing procedure. First, the open-set segmentation method processes the input image. Afterward, the likelihood scores produced by the OSS method (e.g. reconstruction errors, likelihood scores, etc.) are processed using a superpixel segmentation. Every pixel within a given superpixel is assigned its calculated mean score. The final step includes thresholding the predictions according to some criteria – usually set according to quantiles on the scores – to identify unknown superpixels. The *FuSC* box represents the Fusing Superpixels for Improved Semantic Consistency method proposed in this work. The *Segment averaging* diamond represents the averaging processing of the OSS output pixel scores using FuSC superpixels.

Traditional (closed-set) semantic segmentation (SS) for images in general has become a complex, time-consuming, and well-studied task. Many different methods have been developed to solve this problem: Fully Convolutional Network (FCN) [6], U-Net [7], SegNet [8], among others. In recent years, many models have significantly improved the quality of semantic segmentation [9], [10].

It is crucial to improve the semantic and spatial consistency of the segmentation predicted by the SS model to enable the use of such models in practical situations. According to Sekkal et al. [11], the regions need to remain coherent with the original content and a better detection of contours leads to an efficient pseudo-semantic representation.

In closed-set SS, both the training and test data share the same label and feature spaces. However, in more realistic scenarios, unseen classes may appear in the deployment phase of the model. This is the case for most real-world applications, such as autonomous vehicles, medical diagnosis or treatment, and remote sensing tasks. The existence of unknown classes undermines the robustness of the existing closed-set methods, as stated by Geng et al. [12].

For the last couple of years, methods that extend traditional closed-set SS were proposed to automatically recognize samples from unseen classes. This new task, named Open Set Segmentation (OSS), must be able to correctly segment pixels of classes available during training (Known Known Classes – KKCs), while also being able to recognize unknown pixels that come from classes that were not present during training (Unknown Unknown Classes – UUCs) [13].

The goal of the current paper is to improve the semantic consistency of previously obtained results from OSS algorithms for remote sensing images. For this, we introduce in this work a novel method called Open Gaussian Mixtures (OpenGMM), based on previously proposed frameworks for Open Set Recognition (OSR) [13], [14]. Da Silva et al. [13] employed Principal Component Analysis (PCA) in the OpenPCS method as a generative model, assuming it would be sufficient for the representation of the data. As data from known classes in the real world can rarely be correctly represented by unimodal distributions, our proposed approach adopts a Gaussian Mixture Model (GMM) [15], [16] instead of PCA, allowing for modeling KKCs with multiple modes and clusters in the feature space, aiming to improve out-of-distribution (OOD) identification.

Simpler methods based on approximations of statistical distributions followed by thresholding, such as OpenPCS or OpenMax show poor performance in correctly delineating object boundaries, making it difficult to use this type of OSS approach in real-world scenarios. To address this issue and improve the final prediction, we explored how to post-process using Superpixel Segmentation (SPS) algorithms. Figure 1 shows an overview of the post-processing approach proposed in this work. First, the superpixel algorithm and the OSS method process the input image. Then, the outputs are combined by averaging the open-set scores for each superpixel, producing a final open-set segmentation prediction (**FuSC refined open-set prediction**) with better semantic consistency. The lack of semantic consistency produced by methods such as OpenPCS, especially in object borders, can be seen in **Open-set prediction**.

Superpixels are a homogeneous and contiguous group of pixels in an image, extracting perceptually relevant regions [17]. Superpixel algorithms yield an over-segmentation of an image and have played a relevant role in the traditional segmentation pipeline. According to Lin et al. [18], they can be considered low-level image information subdivisions. As stated by [17], [19], [20], the use of superpixels brings several advantages: it reduces the size of the classification problem, since one cluster represents many pixels; it allows for a richer feature representation; and it produces homogeneous regions with additional semantics despite being an over-segmentation.

Our framework employs a novel superpixel merge procedure using Malahanobis distance [21], called **Fu**sing **S**uperpixels for **S**egmentation (FuSC). A graphical illustration of the inner workings of our pipeline can be seen in Figure 1,

We tested our approach in OSS using three distinct algorithms from the literature: Open Principal Component Scoring (OpenPCS) [13], Conditional Reconstruction for Open-set Semantic Segmentation (CoReSeg) [22], and the newly proposed OpenGMM. Our pipeline seeks to improve the semantic consistency of the results to make OSS models more suitable for practical use. Our contributions can be summarized as:

- The proposal of OpenGMM, a novel method aiming to improve upon the previously proposed OSS framework by Oliveira et al. [13];
- The proposal of a superpixel post-processing method that yielded results with superior semantic consistency and improved Receiver Operating Characteristic (ROC) metrics in all tested scenarios;
- A novel superpixel segmentation fusion procedure using Malahanobis distance [21];
- State of the Art OSS results for the Vaihingen and Potsdam datasets[1] using our post-processing segmentation technique.

This manuscript is organized as follows: Section I presents the related work on superpixels for semantic segmentation; Section II describes the proposed methods; Section III introduces the experimental setup used for this work, along with the employed datasets and metrics; Section IV presents our ablation study executed on the Vaihingen dataset; Section V presents the final OSS results obtained on Vaihingen and Potsdam datasets; and, finally, Section VI discusses the conclusions obtained from our experiments.

A. SEMANTIC CONSISTENCY IN SEGMENTATION

Semantic consistency is rarely explicitly addressed in semantic segmentation papers. However, due to the inherent difficulties of OSS scenarios in comparison to traditional supervised SS, semantic consistency is a rather more challenging aspect when there are unknown classes during deployment. In the following lines, we present an overview of the few existing trends in deep semantic consistency.

The method proposed by Ji et al. [23] improved the performance and the spatial consistency of the resulting segmentation for PASCAL VOC 2012, PASCAL-Context, and Cityscapes datasets using an end-to-end trainable network that combines two branches: one for edge detection and one for traditional semantic segmentation.

PixMatch [24] uses heavy augmentation and a loss term composed by a summation of two cross-entropy terms: the first loss term is standard for SS, while the second term is computed over a slightly perturbed image and mask. The new loss enforces the notion of smoothness in the target domain to enhance intra-object segmentation consistency.

Pixelwise Contrast and Consistency Learning (PiCoCo) [25] seeks consistency in closed-set semantic segmentation using a joint loss function that is the summation of a supervised loss term, a contrastive loss term and a consistency loss term. The supervised loss term is composed of a Cross-Entropy and a Dice loss; for the contrastive loss term, a selection of positive and negative samples enforce the model to improve its generalization capabilities; the consistency loss term consists of a summation of cross-entropy and a dice loss of heavily augmented pairs of input and labels to enforce semantic consistency and robustness to the learning process.

The post-processing proposed by Ratajczak et al. [26] combines an unsupervised colorization and a deep edge superpixel segmentation to enhance the semantic segmentation of panchromatic aerial images. The authors propose to assess if applying a colorization algorithm could improve the strength of the pairwise potentials used in a conditional random field (CRF) post-processing. This method computes intermediate Deep Edge Superpixels using Watershed [27] in the intermediate activation maps obtained before each pooling layer. The method uses the generated superpixels

[1] https://www.isprs.org/education/benchmarks/UrbanSemLab/

with the mean value for intensity and a CRF to improve the final semantic segmentation consistency.

The approach proposed by Zhang et al. [28] uses supervoxels to improve the consistency of semantic segmentation. The method used a 3D-CNN (3D Convolutional Neural Network) to learn discriminative hierarchical features from spatiotemporal volumes.

Our work introduces a superpixel post-processing for OSS that improved the semantic consistency of the final segmentation prediction for all tested scenarios. We also propose a novel superpixel segmentation method, called FuSC, that benefits from merging different input segmentations and produces final superpixel segmentation with better results in most tested scenarios compared to the same post-processing with base superpixel algorithms.

B. OPEN-SET SEMANTIC SEGMENTATION

According to Scheirer et al. [29], an open-set scenario happens when the model is not fed all classes during training, allowing unknown samples to appear in the prediction phase. This definition can be applied to each pixel in an image, extending the traditional semantic segmentation to OSS.

OSS has only a handful of published works that use neural networks. To better understand the OSS literature, we will show the first attempts to perform open-set recognition from a neural network proposed by Bendale and Boult [30]. The first natural approach was to apply a threshold on the final output probability, identifying low probabilities as unknown. But, experiments showed that this strategy produced poor results and was not well suited to handle the task. The second approach was called OpenMax, which introduced a new final layer to replace the traditional softmax during deployment. OpenMax adds an Unknown output class and estimates the probability of the input images to each of the $C + 1$ classes.

Based on OpenMax [30], OpenPixel [31] uses a patch-wise strategy to classify the central pixel. OpenPixel is extremely inefficient during the testing phase since each pixel in an image generates a patch to be classified by the network. The fully convolutional counterpart to OpenPixel, named OpenFCN, was proposed by Oliveira et al. [13]. OpenMax-based methods proved to have their effectiveness severely limited in segmentation, resulting in false positive OOD pixel predictions mainly at object boundaries, where the activations of the last layers are affected by the presence of neighboring objects.

Given the limitations of OpenPixel and OpenFCN caused by the last layers' activations, Oliveira et al. [13] proposed using intermediate multiscale features from the closed-set FCNs coupled with low-dimensional principal component analysis scores for OSS. The method, called Open Principal Component Scoring (OpenPCS), achieved consistently better results on the Vaihingen, Potsdam, and Geoscience and Remote Sensing Society (GRSS) 2018 Data Fusion Challenge [32] datasets in comparison to OpenPixel and OpenFCN.

Cui et al. [33] proposed a nonparametric statistical OSS method that employs the Mann-Whitney U test on a closed-set segmentation output to determine the existence of unknown classes in each image. Furthermore, it uses an adaptive threshold that identifies which pixels are unknown.

Proposed by Cen et al. [34], an open-world semantic segmentation system used prototypes for the known classes and a Deep Metric Learning Network (DML-Net) as a feature extractor. This work used a Euclidean distance-based probability loss to the predefined prototypes to identify unknown pixels.

Nunes et al. [22] proposed a fully convolutional end-to-end CoReSeg that tackles the OSS using two network branches: a traditional closed-set segmentation branch and a class-conditional reconstruction of the input images according to their pixel-wise mask, using the reconstruction error from the conditional reconstruction branch to identify the OOD pixels.

The Generalized Open-set Semantic Segmentation (GOSS) method proposed by Hong et al. [35] employs two network branches trained together in parallel: the first branch performs a SS for known classes and identifies unknown pixels using Deep Metric Learning (DML) [34]; the second branch is a pixel clustering that ignores the known classes producing a new segmentation mask for the image. At last, the fusion phase uses the pixels defined as unknown and the clustering to identify different objects in the unknown areas.

This work proposes a novel OSS method called Open Gaussian Mixture of Models (OpenGMM) that modifies the OpenPCS framework and improves quantitatively and qualitatively the final results.

C. SUPERPIXEL SEGMENTATION

SPS has been an active research area for decades, with many methods proposed. Superpixels are groups of contiguous pixels in an image, clustered according to some homogeneity measure. As spatiality is crucial to any SPS, neighboring superpixels should be perceptually different. Nevertheless, non-neighboring superpixels may have similar values and shapes. As examples of proposed techniques in the last two decades: Felzenszwalb and Huttenlocher [17]; Quickshift [36]; TurboPixels [37]; Entropy Rate Superpixel (ERS) [38]; Simple Linear Iterative Clustering (SLIC) [39]; Generative Superpixel Method [40]; Eikonal-based [41]; Superpixels Extracted via Energy-Driven Sampling (SEEDS) [42]; Linear Spectral Clustering (LSC) [43]; Waterpixels [44]; Boundary-Aware Superpixel Segmentation (BASS) [45]; cale-adaptive superpixels (SAS) [46]; Self-Organization-Map Superpixels (SOMS) [20]; content-based [47]; Superpixel Spatial Intuitionistic Fuzzy C-Means Clustering (SPFCM) [48].

Among all possible choices of SPS algorithms, we chose three algorithms that have fundamentally different strategies to generate the superpixels: SLIC [39], Quickshift [36] and Felzenszwalb and Huttenlocher [17]. In the following paragraphs, we briefly present these three superpixel algorithms.

Simple Linear Iterative Clustering (SLIC) [39] algorithm groups pixels into perceptually meaningful contiguous regions using a K-means algorithm [49]. It starts with predefined n centers uniformly distributed in the image. Then, it adjusts the centers' positions to the local minimum of the pixel intensity gradient to avoid centering the superpixel in an edge.

Quickshift [36] is a fast mode seeking algorithm. Initially, each pixel is a superpixel, and then neighboring pixels are merged into the same cluster according to a predefined radius distance. This method does not force pixels to be spatially close to each other, which generates highly homogeneous superpixels of different sizes and shapes.

Felzenszwalb algorithm [17] is a graph-based superpixel segmentation algorithm where each vertex represents a pixel, and each selected edge has some dissimilarity measured as its value. Every pixel in the image is a vertex in the graph, but only some edges are added according to a defined neighboring criterion (e.g. K-nearest neighbors) to guarantee the intended complexity for the algorithm ($O(m \log n)$, where m is the number of edges and n the number of vertices).

In this work, we proposed an OSS superpixel post-processing and a new superpixel merge procedure that improved the quantitative results in all tested scenarios by a large margin. The final refined open-set prediction presented a segmentation perceptually closer to the ground truth.

II. IMPROVING OSS SEMANTIC CONSISTENCY

In the following section, we present our proposed methodology for improving the semantic consistency of OSS. Firstly, in Section II-A, we present an extension of the OSS framework proposed by Oliveira et al. [13], replacing the unimodal dimensionality reduction of PCA with a GMM [15] capable of opening closed-set pretrained segmentation networks with a better segmentation quality. GMM's multimodal data representation should be better suited for modeling real-world pixels that may not conform to the PCA's unimodal data representation, as illustrated in Figure 2. In Section II-B, we propose a superpixel post-processing for generic OSS methods capable of improving quantitative segmentation metrics, as well as qualitative semantic consistency. At last, in Section II-C, we introduce a novel superpixel merge method that uses the Malahanobis [21] distance to merge neighboring superpixels enforcing a minimum pixel count for each segment.

A. OPEN GAUSSIAN MIXTURE OF MODELS

Open Gaussian Mixture of Models (OpenGMM) processes intermediate feature maps with the last layers' activation maps of a deep neural network. Combining the activations from earlier layers with final layers produces a tensor that fuses low and high-semantic-level information. The concatenated tensor may have hundreds or thousands of channels, which are known to contain redundant information [50], [51]. OpenGMM handles the concatenated tensor size and redundancy by fitting a GMM on each known-class distribution. Each GMM model computes a score tensor with the log-likelihood values for all pixels, which allows for the computation of a final score tensor by combining all GMM scores with the closed-set prediction. All pixels below a certain threshold in the final score are identified as unknown.

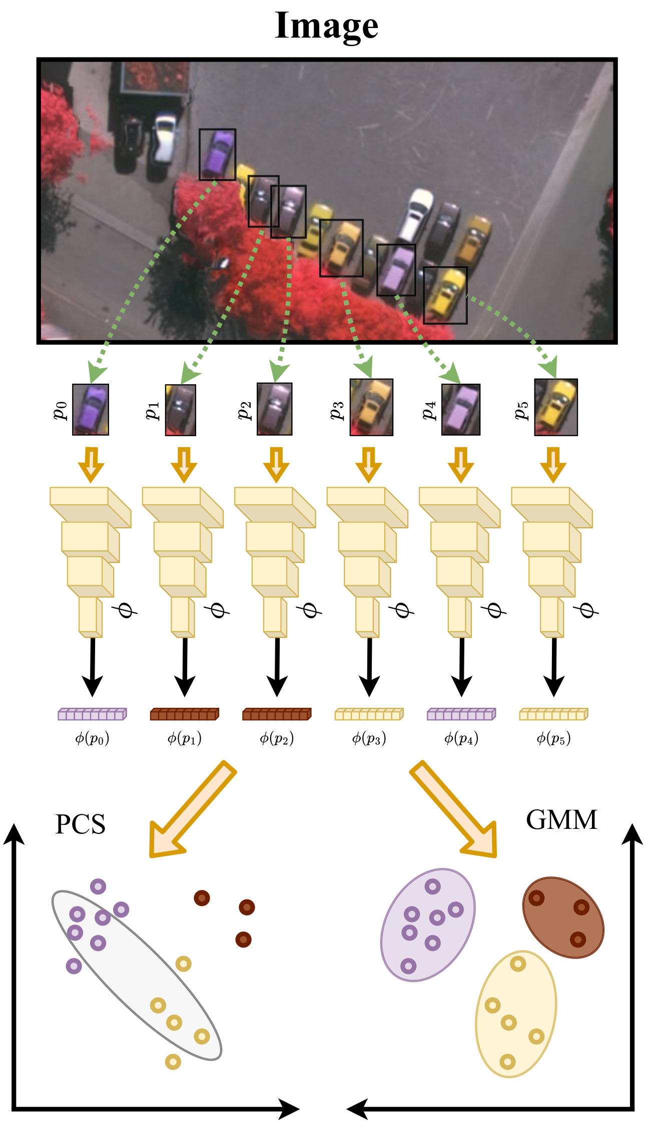


FIGURE 2. This illustration shows how different objects from the same class can be better represented by distinct distributions. Due to its multimodal representation capability, GMM is better suited for representing real-world data than Principal Component Scoring [13].

We performed tests with three different backbones as the closed-set segmentation method: Densely Connected Convolutional Networks (DenseNet-121, shortened as DN121) [51], Wide Residual Networks (WideResNet-50, shortened as WRN50) [52] and U-net [7].

Readers should notice that adapting any pretrained closed-set semantic segmentation network to the OpenGMM frameworks is relatively quick and straightforward and does not require retraining the neural network. The only trainable component in our framework is the GMM to fit into the data, which is considerably faster than retraining a neural network. The plug-and-play characteristic of the method is a great advantage when considering the problem of adapting the solution to real-world applications and novel domains.

B. IMPROVING SEMANTIC CONSISTENCY WITH SUPERPIXELS

Superpixels are commonly used in imaging processing as a compact representation of the image, as part of the processing pipeline, or even as post-processing refinement of an output. Many methods use superpixels as part of the segmentation pipeline [23], [24], [25], [26], [28].

To achieve better semantic consistency, we choose to apply the superpixel segmentation to the scores produced by the OSS methods. The output of the OSS method produces a score tensor of dimensions $H \times W \times 1$, where H and W are the height and width of the input image. We then apply SPS to the output tensor, and all pixels of each superpixel are set to the average score value of the segment they belong to. Algorithm 1 details the use of superpixel over-segmentation in the final step of the open-set segmentation just before the open-set pixel identification step.

Algorithm 1 The Output of the OSS Method Is Post-Processed Using the Superpixels Segmentation. All Pixels of a Given Segment Assume the Mean Value of All Pixels in That Segment

Require: scores ▷ pixelwise array
Require: segments ▷ list of segments

procedure post_process(*scores*, *segments*)
    pred = zeros(scores.size)
    for *segment* ∈ *Segments* **do**
        pred[segment] = mean(scores[segment])
    end for
    return pred
end procedure

Since our post-processing scheme is agnostic to the choice of superpixel segmentation algorithm, we evaluated SLIC [39], Quickshift (QS) [36] and Felzenszwalb (FZ) and Huttenlocher [17]. To reinforce the idea that post-processing is robust and can deliver good results regardless of the superpixel generation algorithm, we chose algorithms with different generation strategies, assumptions, and internal metrics.

The remaining steps of the open-set segmentation process were kept as in each OSS method. Superpixel over-segmentations are homogeneous and tend to respect object borders. Applying the superpixels to the score image smooths the segmented areas, aiding the OSS algorithm in avoiding errors due to OOD pixels within the segmented objects, which is a common source of segmentation errors.

Each superpixel algorithm has its own generation characteristics, and the final segmentation reproduces these

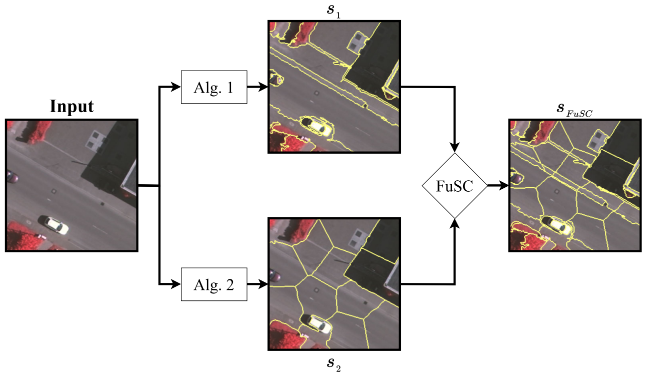


FIGURE 3. Illustration of the workflow to merge two different superpixel segmentations. First, the input image x is processed by 2 different superpixel segmentation algorithms (Alg. 1 and Alg. 2). Afterwards, the generated segmentations s_1 and s_2 are merged into the final segmentation s_{FuSC} using the merging procedure described in Algorithm 2.

particularities in the results. We can see in Figure 4 an illustrative example of two SPS methods that present different characteristics and may represent better different scenarios. In this example, the SLIC algorithm could better represent textures, while the FZ algorithm could better identify borders. None of the single superpixel segmentation algorithms could represent the underlying image properly.

Algorithm 2 Pseudo-Algorithm for the FuSC Procedure and the Auxiliary Procedure of Joining Segmentations. The Complexity of the Procedure Is Pseudo-Polynomial With Respect to the Number of Pixels in the Image and the Minimum Size of the Superpixel (Appendix B)

Require: seg1, seg2 ▷ list of segments

procedure join_segmentations(*seg*1, *seg*2)
    joint = []
    for *s*1 ∈ *seg*1 **do**
        ▷ Selecting s2 ∈ seg2 where s2 ∩ s1 ≠ ∅
        for s2 ∈ seg2.overlap_segments(s1) **do**
            overlap_area = *s*1 ∩ *s*2
            joint.add_new_segment(overlap_area)
        end for
    end for
    return joint
end procedure

procedure FuSC(*seg*1, *seg*2)
    joint = join_segmentations(seg1, seg2)
    for *s* ∈ *joint* **do**
        if *s.size* < *min_size* **then**
            closest = closest_neighbor(s, joint)
            joint = merge_segments(joint, s, closest)
        end if
    end for
    return joint
end procedure

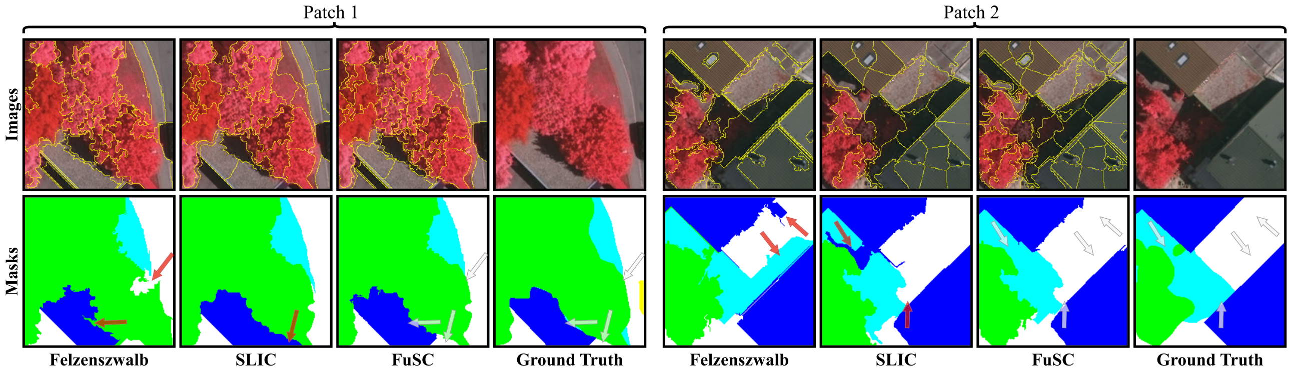


FIGURE 4. Comparison of the resulting segmentation from two SPS algorithms (Felzenszwalb [17] and SLIC [39]) and our proposed fusion algorithm, FuSC. The first row shows the input image superimposed with the superpixel segments and the second row depicts the closer class fit of each segment according to the real labels. Red arrows indicate areas where class boundaries failed when using one single SPS algorithm, while gray arrows point to these same regions fixed after applying the FuSC algorithm.

C. FUSING SUPERPIXELS FOR IMPROVED SEMANTIC CONSISTENCY

All superpixel segmentation algorithms share the same main goals: generate homogeneous areas and respect boundaries between objects. SPS algorithms aim to minimize the intracluster/intrasegment variance while maximizing the intercluster/intersegment variance.

Different superpixel segmentation algorithms use distinct procedures and premises to produce the final segmentation. From distinct superpixel segmentation construction processes, each over-segmentation fails and succeeds in different ways to achieve the intended representation.

The FuSC procedure fuses input segmentations from multiple types of superpixel generation algorithms. Using distinct family algorithms allows FuSC to take advantage of the different generation characteristics, amplifying the strengths and mitigating the weaknesses of each method. Figure 4 exemplifies how each segmentation relates to the ground truth and compares to FuSC joint segmentation. We can observe through the qualitative result that the FuSC improved the representation concerning the ground truth.

Figure 3 illustrates the merging of two different superpixel segmentations, showing that FuSC respects segmentations' borders, and each joint segment can better represent the underlying region. FuSC is agnostic to the SPS algorithm, being applicable to any set of distinct superpixel segmentations. However, in practice, using more than two segmentations yields exceedingly small segments, motivating our experiments to focus only on pairs of segmentations. Having exceedingly small merged superpixels prevents the joint segmentation from taking advantage of the distinct generation strategies, favoring the prevalence of the Mahalanobis distance merging procedure over the original segmentations.

The first step of the fusion procedure is to generate unique segments by superposing two different segmentations, running the following steps:

1) generate a new segmentation from the intersection of the input segmentations;
2) ensure that the final superpixel segmentation respects the minimum size for each segment.

The first step have theoretical complexity linear on the number of pixels of the image (O(n), where n is the number of pixels). This initial merging procedure is prone to produce some tiny segments.

To tackle the unwanted small segments side-effect, we use the Mahalanobis distance [21] to fuse the closest neighbor segments until there are no more segments below the specified pixel minimum size. Algorithm 2 details the FuSC procedure.

The final theoretical complexity of the FuSC procedure is $O(n \times minimum\_size^2)$, where n is the number of pixels and *minimum_size* is a constant parameter of the merge procedure. Since the *minimum_size* is constant, the final complexity of FuSC is O(n). The code in Python with the depiction of the complexity analysis is in Appendix B. This theoretical complexity makes the use of FuSC viable in production scenarios.

III. EXPERIMENTAL SETUP

We used PyTorch [53] to implement all neural network models and an NVIDIA Titan X with 12GB of memory. SPS algorithms were implemented using the *scikit-image*[2] library and we used the GMM implementation from *scikit-learn*[3]. The official implementation for FuSC is publicly available to encourage reproducibility[4].

We tested OpenGMM with 4, 8, and 16 components, resulting in minimal performance differences across this range of values. Thus, for simplicity, all OpenGMM's results reported in this section used 4 Gaussian components, OpenPCS was trained with 16 components following the experimental setup of [13].

[2]https://scikit-image.org/
[3]https://scikit-learn.org/
[4]https://github.com/iannunes/FuSC

TABLE 1. Results aggregated by type of superpixel generation. Two types of generation were used, "single" and "FuSC". "Single" stands for a generation using one superpixel method alone, while "FuSC" stands for the procedure presented in Section II-C. Columns min, avg. and max stand for the minimum, average and maximum AUROC value among all superpixel configurations for each type of generation. The last block is the is the average value among all tested UUCs.

Type	UUC: imp. surf.			UUC: building			UUC: low veg.			UUC: high veg.			UUC: car			Average		
	min	avg.	max	min	avg.	max	min	avg.	max	min	avg.	max	min	avg.	max	min	avg.	max
single	.860	.898	**.912**	.934	.956	**.961**	.702	.730	**.740**	.824	.873	.886	.709	.872	**.918**	.830	.866	.878
FuSC	**.903**	**.907**	.911	**.957**	**.960**	**.961**	**.729**	**.735**	**.740**	**.878**	**.884**	**.887**	**.813**	**.884**	.913	**.860**	**.874**	**.888**

A. DATASETS AND EVALUATION PROTOCOL

We employed the Leave One Class Out (LOCO) protocol used by Oliveira et al. [13] to emulate an open-set scenario on the two selected datasets. The LOCO protocol splits the known classes and selects one class at a time to be ignored during training allowing open-set methods to be evaluated over the hidden class. Thus, we only backpropagate the loss of pixels from the known classes, ignoring the background, borders, miscellaneous and unknown classes.

We selected the Vaihingen and Potsdam datasets from the International Society for Photogrammetry and Remote Sensing (ISPRS[5]) for our experiments. The datasets used have already had benchmarks established for OSS scenarios by published papers [13], [22], [31].

Vaihingen images present a 9cm/pixel spatial resolution, varying from 2000 to 2500 pixels per axis, while Potsdam samples have a 5cm/pixel spatial resolution and 6000 × 6000 pixels each. Both datasets contain six (6) known known classes (KKCs): impervious surfaces, buildings, low vegetation, high vegetation, car, and miscellaneous. Among the KKCs, we removed the miscellaneous class from our experimental procedure since it is mainly comprising areas that exhibit image acquisition noise and objects unimportant for practical remote sensing applications. We used the same bands employed in previous works on OSS: Near Infra-Red (NIR), Red (R), Green (G), and the Digital Surface Model (DSM).

We separate the datasets into three sets each: training, validating, and testing. For the Vaihingen dataset, the selected patches were: 1, 3, 5, 7, 13, 17, 21, 26, 32, and 37 for training; 11, 15, 28, 30, and 34 for testing; and 23 for validation. For the Potsdam dataset, the selected patches were: 2_10, 2_13, 2_14, 3_10, 3_12, 3_13, 3_14, 4_11, 4_12, 4_13, 4_14, 4_15, 5_10, 5_12, 5_13, 5_14, 5_15, 6_8, 6_9, 6_10, 6_11, 6_12, 6_13, 6_15, 7_7, 7_9, 7_11, 7_12 and 7_13 for training; 2_11, 2_12, 4_10, 5_11, 6_7, 7_8 and 7_10 for testing; and 3_11 and 6_14 for validation.

B. BACKBONES

All OSS models use closed-set segmentation backbones as a starting point to identify the OOD pixels. Relying on previous results [13], [31], we used three of the best-performing backbones for our experiments: U-Net [7], WRN50 [52] and DN121 [51]. For CoReSeg [22], we used only U-Net since it naturally matches the architectural constraints of this method.

[5] https://www.isprs.org/education/benchmarks/UrbanSemLab/

C. METRICS

We use the Receiver Operating Characteristic (ROC) curve and the Area Under the ROC (AUROC) improvements as evidence that the proposed methods improve OOD recognition. We employed Cohen's kappa score [54] (κ) to measure the performance for known and unknown classes at the same time. The use of κ allows for the assessment of the reliability of the process, since it measures the agreement of the methods with the ground truth. We used the metrics to compare the predictions with and without superpixel post-processing.

D. POST-PROCESSING

As we used two distinct families of techniques, we have different meanings for the scores: OpenGMM and OpenPCS scores are log-likelihood values for each pixel, and CoReSeg scores are the minimum reconstruction error across the known classes for each pixel.

The post-processing technique can be applied at two distinct points in the pipeline: at the final OSS prediction or at the intermediate scores used to identify the OOD pixels. In the first, with the OSS prediction, we use the superpixels to perform a majority class vote detecting the predominant class. In the second, we compute the mean or median of the scores in each superpixel to apply the threshold identifying the OOD superpixels.

E. SUPERPIXEL CONFIGURATIONS

To evaluate the effectiveness of FuSC, we choose three completely different superpixel generation algorithms as our base algorithms: SLIC [39], Quickshift (QS) [36] and Felzenszwalb (FZ) and Huttenlocher [17]. The three selected algorithms use completely distinct ideas to produce the superpixels, and the final segmentations have segments with different sizes, structures, and shapes.

We conducted experiments with 70 different superpixel configurations for the Vaihingen ablation study. We selected the six (6) best configurations on Vaihingen to run in Potsdam, due to the greater computational cost.

The 70 used configurations were of 2 categories:

1) *single* – an execution with different parameters of one of the three selected SPS algorithms;
2) *FuSC* – fusion of two different superpixel segmentations using the proposed procedure.

For the FuSC merging procedure, we tested both mean and median when calculating the distance between neighboring

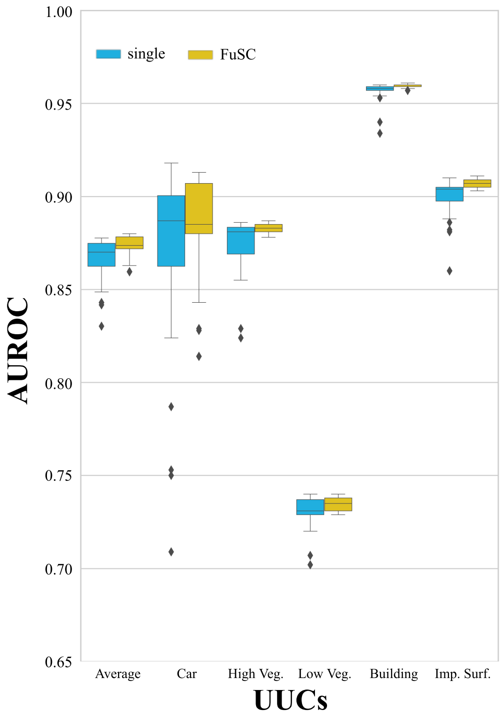


FIGURE 5. Boxplot with the AUROC results for all configurations used for the ablation study for CoReSeg in the Vaihingen dataset. The boxplot shows both single superpixel algorithms (blue) and FuSC (yellow) for individual UUCs and for the average.

superpixel regions. Furthermore, we tested two different minimum sizes for the superpixels: 25 and 50 pixels. By default the SPS algorithms use the mean to represent each segment, or to define the SPS, so we tested the median to discover if it could better represent the image bands.

In the Appendix A, we list the 11 superpixel generation configurations reported in this work in Sections IV and V. As a results of the Section IV, FuSC's result reported in Section V used the *mean* value for the scores of each superpixel and the minimum superpixel size of 50 pixels.

IV. ABLATION STUDY

Besides the 70 superpixel generation configurations mentioned in Section III, we applied the superpixels in four (4) distinct manners, adding up to 280 different superpixel post-processed results to evaluate. We performed the ablation study for OpenGMM due to the computational cost of running 280 tests for each method. To evaluate the post-processing in a distinct methods family, we also extended the ablation study for the Vaihingen dataset using CoReSeg [22], which uses U-net as a fixed architecture.

We split the following section into two sets of experiments: 1) Superpixel generation (Section IV-A), and 2) Post-Processing for OSS (Section IV-B).

A. SUPERPIXEL GENERATION STRATEGY

In this section, we intend to evaluate and compare the single superpixel algorithms with the FuSC procedure identifying the strengths and weaknesses between the methods.

Table 1 and Figure 5 present the worst, best, and average results aggregated by generation strategy for each UUC and for the average. We can observe that the worst and best results produced by FuSC are closer, achieving considerably stabler results in all scenarios, while also producing better average results. Figure 5 shows comparatively how FuSC results are closer than SPS results in all cases.

Selecting the correct parameters is critical when using superpixel generation algorithms to achieve a suitable image representation. The construction of FuSC combines different SPS algorithms, with the results suggesting that the selection of parameters becomes less relevant to the process, validating the conjecture that combining 2 different over-segmentations generates a more reliable result.

Figure 6 presents the minimum, average, and maximum results for the two distinct generation strategies. The error bars show the confidence intervals given by a paired two-tailed t-Student test with $p \leq 0.05$ across all five (5) tested scenarios. We can observe the better comparative performance of FuSC in all three tested cases.

The ablation study indicated that the results using the mean on all the superpixels pixels to represent them produced slightly better results. The ablation results showed that using 50 pixels as the minimum size for the superpixels also produced slightly better results. We can see in Table 8 in Appendix A a list with the best 20 results among all executed tests for this ablation study.

The collected results can be used as evidence suggesting that FuSC is more stable, less sensitive to parameter selection, and produces better results on average.

B. POST-PROCESSING FOR OSS

The first ablation study pointed out that FuSC seems to be better comparing all gathered results. In this section, we study where to apply the post-processing and how represent the superpixel.

Regarding the use of superpixel segmentation to improve the results of the OSS methods, we tested four (4) different strategies (a pair of strategies with two possible settings each):

1) regarding to where to apply the superpixel segmentation:
   a) applying to the final closed-set segmentation;
   b) applying only to the scores/reconstruction errors;
2) regarding to how to represent the superpixel:
   a) using the mean to represent the score/reconstruction error for the entire superpixel;
   b) using the median to represent the score/reconstruction error for the entire superpixel.

Table 3 presents the results aggregated by the strategies used in the OSS post-processing. We observed the same performance applying the produced superpixels on the final tensor before predicting the OSS segmentation and

TABLE 2. For each combination of backbone and OSS method the table shows the classwise AUROC and average (Avg.) AUROC in Vaihingen and Potsdam. Numbered columns stand for: 0 - Impervious Surfaces, 1 - Building, 2 - Low Vegetation, 3 - High Vegetation and 4 - Car.

Backbone	OSS method	Vaihingen UUCs 0	1	2	3	4	Vaihingen Average	Potsdam UUCs 0	1	2	3	4	Potsdam Average
DN121	OpenGMM	**.872**	**.936**	**.646**	.688	**.687**	$\mathbf{.7658 \pm .129}$	.829	**.907**	**.642**	.553	.866	$\mathbf{.7594 \pm .154}$
DN121	OpenPCS	.860	.925	.643	**.708**	.650	$.7572 \pm .128$	**.841**	.901	.546	**.569**	**.882**	$.7478 \pm .175$
U-Net	CoReSeg	.884	**.934**	**.710**	**.867**	**.854**	$\mathbf{.8499 \pm .084}$	.824	**.901**	**.698**	**.631**	**.768**	$\mathbf{.7644 \pm .106}$
U-Net	OpenGMM	**.908**	.787	.522	.846	.601	$.7328 \pm .165$	**.870**	.856	.371	.569	.858	$.7048 \pm .226$
U-Net	OpenPCS	.899	.799	.508	.843	.538	$.7174 \pm .181$	.875	.859	.423	.531	.835	$.7046 \pm .212$
WRN50	OpenGMM	**.885**	**.911**	.611	.648	.619	$\mathbf{.7348 \pm .150}$	.851	.870	.403	.453	.824	$.6802 \pm .231$
WRN50	OpenPCS	.854	.873	**.628**	**.658**	**.622**	$.7270 \pm .126$	**.853**	**.880**	**.424**	**.462**	**.840**	$\mathbf{.6918 \pm .228}$

TABLE 3. The table shows the AUROC results by how the post-processing was used and the used value for the superpixel. The column "Superpixel All" indicates if the superpixel segmentation was applied to the final predicted segmentation or the OSS scores tensor instead, and the "Value Used" column shows which value represents the entire superpixel.

UUCs Imp. Surf.	Building	Low Veg.	High Veg.	Car	Average	Superpixel All	Value Used
.903	.958	**.737**	**.881**	**.897**	$\mathbf{.8752 \pm .083}$	Yes	Mean
.906	**.959**	.730	.880	.865	$.8676 \pm .085$	Yes	Median
.903	.958	**.737**	**.881**	**.897**	$\mathbf{.8752 \pm .083}$	No	Mean
.906	**.959**	.730	.880	.865	$.8677 \pm .085$	No	Median

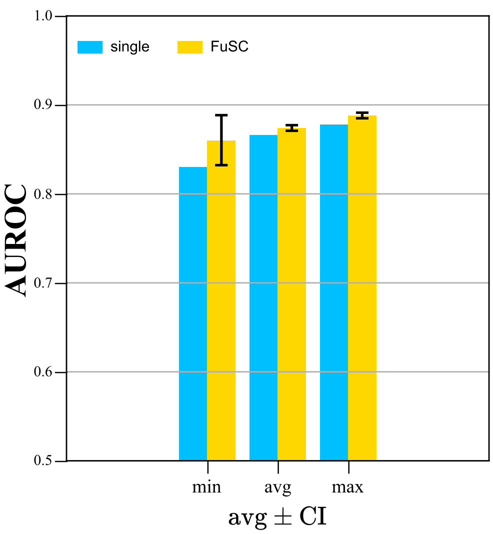


FIGURE 6. Comparison of minimum, average and maximum AUROC for all superpixel configurations generated for the ablation study on CoReSeg [22] with the Vaihingen dataset. The barplot shows single superpixel algorithms (yellow) and FuSC (blue) for the average across all UUCs in the LOCO protocol. Confidence Intervals (CIs) according to a paired two-tailed t-Student test with $p \leq 0.05$ across the five (5) classes are shown as error bars, highlighting the statistical significance of employing FuSC instead of single SPS algorithms. For better visualization of the CIs, we trimmed the lower y-axis to 0.5.

applying it to the OSS scores/reconstruction errors. Using the *mean* value to represent the superpixel produced the best results. As for applying the superpixel only to the scores/reconstruction errors or along the closed-set final tensor, the results show similar performance, implying that the variations are equivalent. Due to the proximity of the results presented in Table 3, we observe that the first and the third lines produced equivalent results with the same scores.

V. RESULTS AND DISCUSSION

A. OPENGMM IN COMPARISON TO BASELINES

Table 2 compares OpenGMM, OpenPCS, and CoReSeg results using the same closed-set backbones without any post-processing. The results in bold are the best ones for each closed-set backbone. Section V-B will present and discuss the post-processing results.

For the Vaihingen dataset, we can observe that in all scenarios, OpenGMM outperformed OpenPCS with the same backbone. The Potsdam dataset yielded mixed results for OpenGMM and OpenPCS, showing overall similar performances. OpenGMM surpassed OpenPCS in 4 out of 6 direct comparisons.

We attribute the improvement shown by OpenGMM over OpenPCS to its multimodal representation capability for modeling real-world data, regardless of the number of KKCs. The worse results in Potsdam are attributed mainly to OpenGMM's poorer performances on two UUCs: Low Vegetation and High Vegetation. The instability of OSS algorithms in these two particular classes is known from previous work, possibly due to the large semantic intra-class variability.

It is worth mentioning that our experimental procedure is not the same used by Oliveira et al. [13]. Therefore, all experiments were re-executed to ensure comparability, and the results presented in this work differ from the original publication of OpenPCS.

A remarkable advantage of OpenGMM is the promptness of the method, meaning that adapting it to other backbones is simple and does not require retraining the neural network. The results achieved by OpenGMM improved the baseline established by OpenPCS in most cases, and the best OpenGMM results are close to CoReSeg's results. Since OpenGMM can benefit from networks producing better

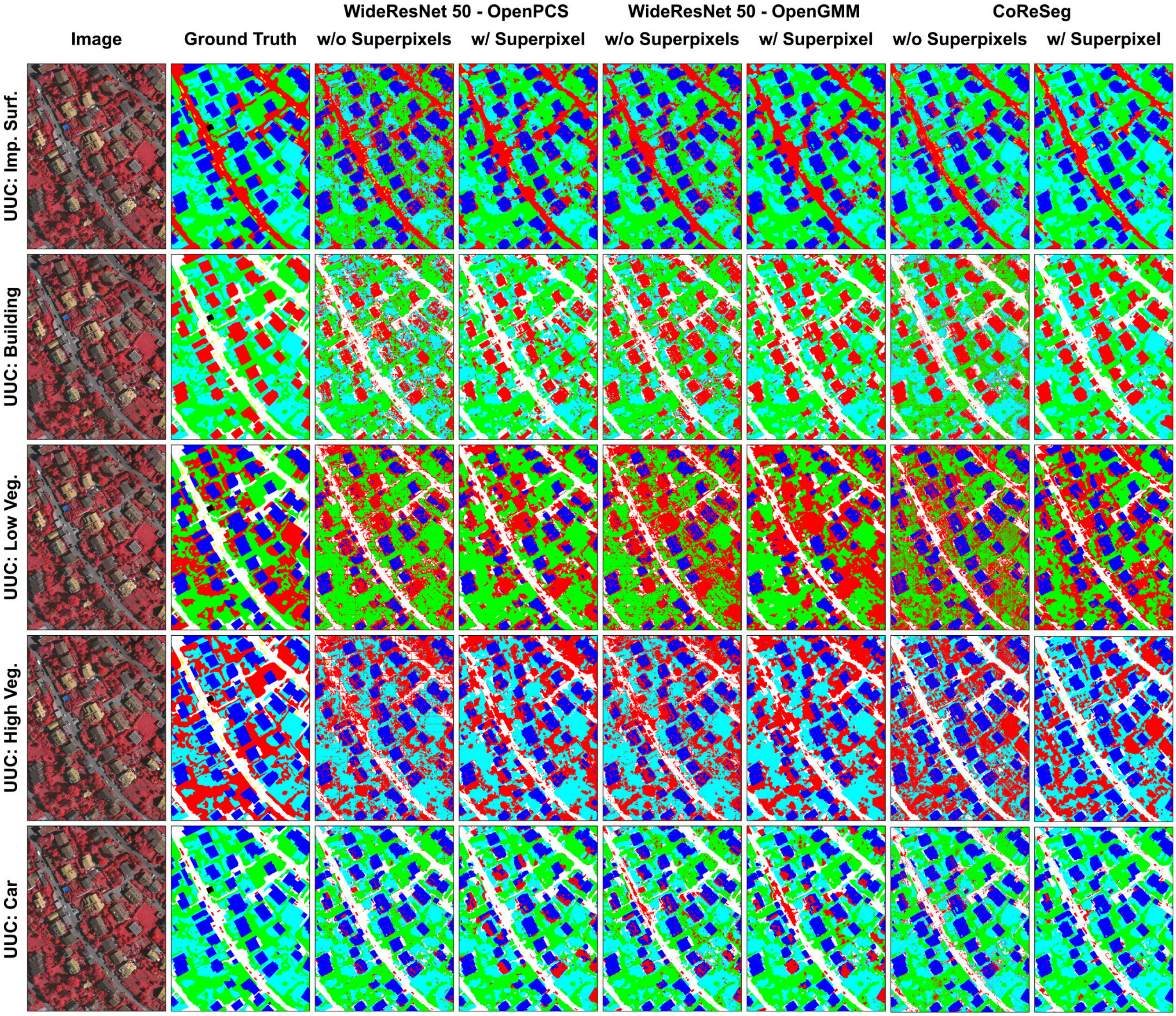


FIGURE 7. **The figure presents qualitative results for an image from the Vaihingen dataset under different settings of UUCs and OSS methods. The *w/ superpixels* shows the results of the best post-processing for the image. The post-processing produced a segmentation reducing the usual mislabeling of unknown pixels (colored in red) and better delineating the boundaries.**

TABLE 4. **The table shows Kappa scores with threshold values varying from 0.9 to 1.0 for the Vaihingen dataset with CoReSeg as the OSS method. The second column indicates the usage of the FuSC post-processing. For the average rows, the † symbol indicates if the results have statistical significance, according to a paired two-tailed t-Student test with $p \leq 0.05$ across the five (5) classes.**

UUC	FuSC	Thresholds										
		0.90	**0.91**	**0.92**	**0.93**	**0.94**	**0.95**	**0.96**	**0.97**	**0.98**	**0.99**	**1.00**
Imp. Surfaces	No	0.678	0.683	0.687	0.692	0.696	0.699	0.701	0.698	0.685	0.564	**0.540**
Imp. Surfaces	Yes	**0.703**	**0.705**	**0.708**	**0.710**	**0.710**	**0.710**	**0.712**	**0.711**	**0.700**	**0.549**	**0.540**
Buildings	No	0.692	0.698	0.703	0.707	0.710	0.709	0.705	0.695	0.674	0.623	**0.504**
Buildings	Yes	**0.744**	**0.746**	**0.748**	**0.749**	**0.747**	**0.742**	**0.733**	**0.717**	**0.689**	**0.626**	**0.504**
Low Vegetation	No	0.587	0.589	0.591	0.592	0.594	0.595	0.597	0.600	0.605	0.611	**0.621**
Low Vegetation	Yes	**0.600**	**0.601**	**0.604**	**0.603**	**0.605**	**0.606**	**0.607**	**0.609**	**0.611**	**0.615**	**0.621**
High Vegetation	No	0.656	0.657	0.658	0.658	0.656	0.653	0.648	0.638	**0.619**	**0.555**	**0.555**
High Vegetation	Yes	**0.683**	**0.684**	**0.683**	**0.683**	**0.681**	**0.676**	**0.665**	**0.645**	0.609	0.549	**0.555**
Car	No	0.677	0.684	0.692	0.701	0.710	0.720	0.732	0.745	0.757	0.769	0.778
Car	Yes	**0.713**	**0.719**	**0.726**	**0.733**	**0.741**	**0.749**	**0.758**	**0.764**	**0.769**	**0.773**	**0.779**
Average	No	0.658	0.662	0.666	0.670	0.673	0.675	0.677	0.675	0.644	0.649	**0.600**
Average	Yes	**0.689** †	**0.691**†	**0.694**†	**0.695**†	**0.697**†	**0.697**†	**0.695**†	**0.689**†	**0.645**	**0.652**	**0.600**

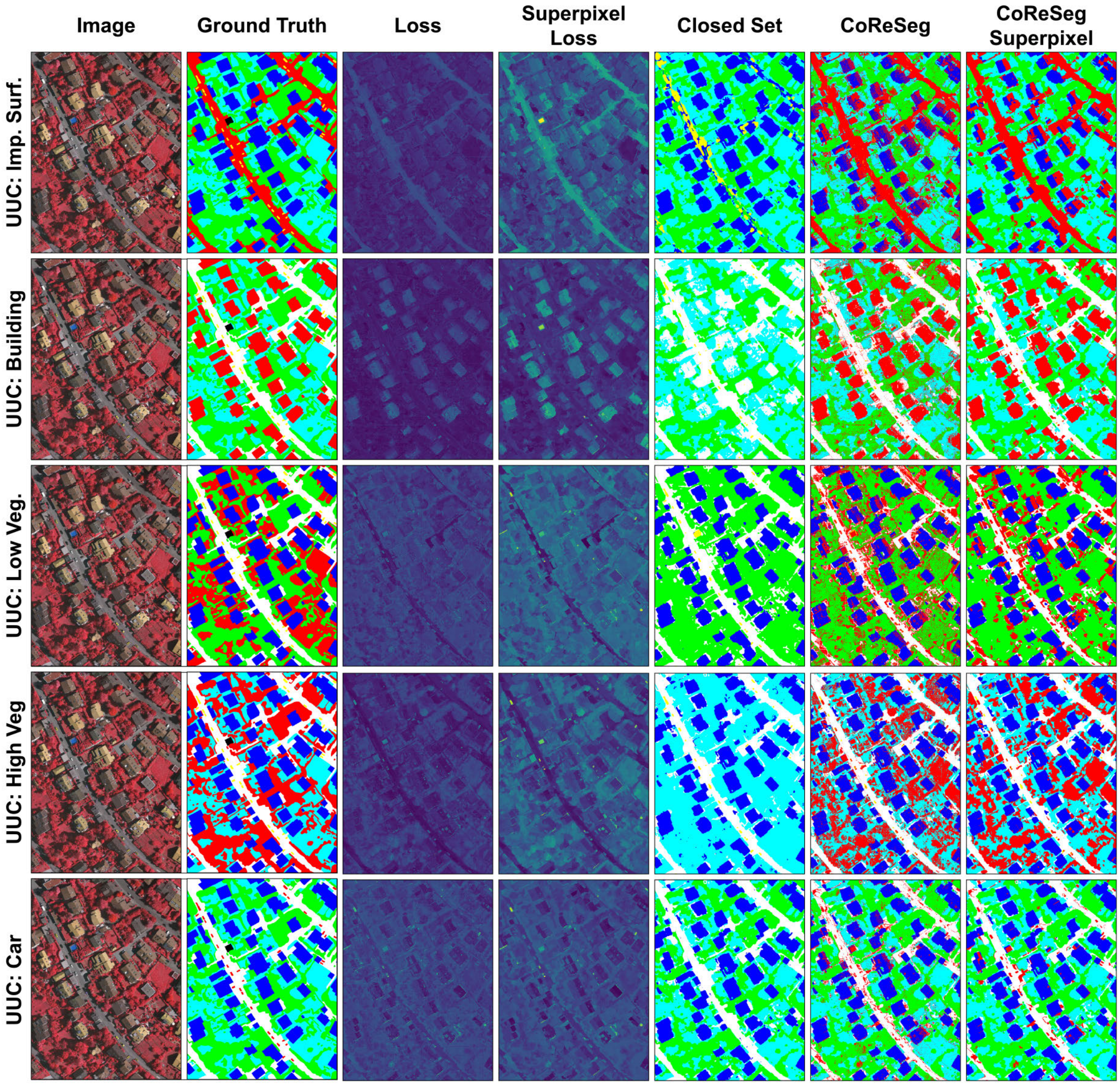


FIGURE 8. The figure shows the OSS predictions obtained using CoReSeg for an image of the Vaihingen dataset with all tested UUCs. The last column presents the qualitative improvement achieved using the proposed superpixel post-processing method. We can also observe the impact of the post-processing on the Reconstruction Loss.

data intermediate representations, further experiments are expected to improve the results.

It is noticeable that OpenGMM obtained its best results using DN121 as its backbone and that CoReSeg only used U-net as its backbone, making the comparison between its best results unfair to CoReSeg. Using DN121 instead of the U-net as a backbone allowed OpenGMM to take advantage of the greater representational ability of the backbone used. However, we can say that CoReSeg also has an advantage over OpenGMM, since the network is trained to identify OOD pixels, while OpenGMM is a module coupled to a network trained only for closed-set segmentation.

B. SUPERPIXEL POST-PROCESSING RESULTS

Superpixel post-processing improved the average AUROC in all tested scenarios, as shown in Tables 5 and 6, and also produced better semantic consistency, as shown in Figures 7 and 8.

Tables 5 and 6 show the AUROC baseline results for the OSS prediction and the improvement observed by post-processing with a single SPS algorithm or using FuSC. The tables present the results for all UUCs and the average of the UUCs, sided by the medium size in pixels of the used superpixel configuration.

We only observed a worsening after post-processing the OSS predictions for the cases with poor closed-set results for a certain UUC. The results suggest that if the baseline OSS prediction presented little semantic consistency, the post-processing could not consistently improve the results. Whenever the baseline AUROC is close to or below 0.50 the post-processing performance becomes unpredictable and may worsen the final result.

TABLE 5. The table presents results for the Vaihingen dataset ordered by the average AUROC for each pair Backbone/OSS Method. Each configuration of the column Superpixel Config. is better detailed in the Appendix A. The column *avg. AUROC* shows the average AUROC among all UUCs; the column *avg. px/seg* shows the average size of the segments produced by the superpixel configuration used to post-process. The † symbol marks when OpenGMM produces better results than OpenPCS with the same Backbone-OSS method. In bold we can see the best results for each backbone.

Backbone	OSS Method	Config.	UUCs					avg. AUROC	avg. px/seg
			Imp. Surf.	Buildings	Low Veg.	High Veg.	Car		
DN121	OpenGMM	-	.872	.936	.646	.688	.687	.7658 ± .129	-
DN121	OpenGMM	fusc01	**.891**†	.946	**.652**†	.698	.689	.7752 ± .133	750
DN121	OpenGMM	single02	.880	**.954**†	.650	.696	**.701**†	**.7762 ± .133**†	491
DN121	OpenPCS	-	.860	.925	.643	.708	.650	.7572 ± .128	-
DN121	OpenPCS	fusc02	.878	.948	.651	**.720**	.636	.7694 ± .135	322
DN121	OpenPCS	single02	.876	.951	.649	.718	.663	.7714 ± .135	491
U-Net	CoReSeg	-	.884	.934	.710	.867	.854	.8499 ± .084	-
U-Net	CoReSeg	single03	.906	.958	.737	.884	.903	.8776 ± .083	476
U-Net	CoReSeg	fusc03	.906	**.959**	**.739**	**.886**	**.910**	**.8800 ± .083**	434
U-Net	OpenGMM	-	.908	.787	.522	.846	.601 †	.7328 ± .165	-
U-Net	OpenGMM	fusc01	**.930**†	.805	.512	.846	.597	.7380 ± .176	750
U-Net	OpenGMM	single01	.929	.802	.519†	.848	.599	.7394 ± .173†	322
U-Net	OpenPCS	-	.899	.799	.508	.843	.538	.7174 ± .181	-
U-Net	OpenPCS	single04	.925	.814	.501	.848	.527	.7230 ± .195	322
U-Net	OpenPCS	fusc02	.925	.813	.502	.852	.526	.7236 ± .196	492
WRN50	OpenGMM	-	.885	.911	.611	.648	.619	.7348 ± .150	-
WRN50	OpenGMM	single01	.904	**.936**†	.622	.665	.618	.7490 ± .165	491
WRN50	OpenGMM	fusc04	**.918**†	**.936**†	.630	.667	**.641** †	**.7584 ± .155** †	698
WRN50	OpenPCS	-	.854	.873	.628	.658	.622	.7270 ± .126	-
WRN50	OpenPCS	fusc04	.896	.915	.628	**.681**	.571	.7378 ± .158	698
WRN50	OpenPCS	single02	.890	.912	**.648**	**.681**	.600	.7462 ± .144	491

TABLE 6. The table shows results for the Potsdam dataset ordered by the average AUROC for each pair Backbone/OSS Method. Each configuration of the column Superpixel Config. is better detailed in the Appendix A. The column *avg. AUROC* shows the average AUROC between the UUCs; the column *avg. px/seg* shows the average size of the segments produced by the superpixel configuration used to post-process. The † symbol marks when OpenGMM produces better results than OpenPCS with the same Backbone-OSS method. In bold we can see the best results for each backbone.

Backbone	OSS Method	Config.	UUCs					avg. AUROC	avg. px/seg
			Imp. Surf.	Buildings	Low Veg.	High Veg.	Car		
DN121	OpenGMM	-	.829	.907	.642	.553	.866	.7594 ± .154	-
DN121	OpenGMM	single03	.852	.914	**.642**†	.559	.871	.7676 ± .154	509
DN121	OpenGMM	fusc01	.848	**.916**†	**.642**†	.567	.887	**.7720 ± .157**†	1006
DN121	OpenPCS	-	.841	.901	.546	.569	.882	.7478 ± .175	-
DN121	OpenPCS	single01	**.866**	.913	.569	.571	.888	.7614 ± .177	876
DN121	OpenPCS	fusc01	.863	.913	.558	**.582**	**.904**	.7640 ± .178	1006
U-Net	CoReSeg	-	.824	.901	.698	.631	.768	.7644 ± .106	-
U-Net	CoReSeg	single05	.860	**.913**	**.738**	.654	.813	.7956 ± .102	1447
U-Net	CoReSeg	fusc01	.866	.911	.733	**.659**	.817	**.7972 ± .102**	1006
U-Net	OpenGMM	-	.870	.856	.371	.569	.858	.7048 ± .226	-
U-Net	OpenGMM	single03	.898	.880	.357	.571	.876	.7164 ± .243	509
U-Net	OpenGMM	fusc01	.895	.878	.361	.578†	**.897**†	.7218 ± .243	1006
U-Net	OpenPCS	-	.875	.859	.423	.531	.835	.7046 ± .212	-
U-Net	OpenPCS	single03	.901	.893	.415	.524	.864	.7194 ± .232	509
U-Net	OpenPCS	fusc04	**.902**	.892	.416	.526	.874	.7220 ± .233	876
WRN50	OpenGMM	-	.851	.870	.403	.453	.824	.6802 ± .231	-
WRN50	OpenGMM	single03	.884	.890	.418	.455	.835	.6964 ± .239	509
WRN50	OpenGMM	fusc01	.880	.890	.413	.457	.853	.6986 ± .242	1006
WRN50	OpenPCS	-	.853	.880	**.424**	**.462**	.840	.6918 ± .228	-
WRN50	OpenPCS	single03	**.888**	**.904**	.423	.457	.856	.7058 ± .243	509
WRN50	OpenPCS	fusc01	.884	**.904**	.419	.457	**.873**	**.7074 ± .247**	1006

However, when the baseline OSS model prediction with no superpixel post-processing yields consistent segmentations, even with border issues or salt-and-pepper artifacts, the superpixel post-processing is quite effective. In other words, assuming that the superpixels are representative, homogeneous, and respect the edges of the image, a better base OSS result allows the superpixel post-processing to correct mistakes inside the superpixels, as these mistakes are usually the minority of pixels. The results suggest that better baseline OSS prediction results would benefit more from the superpixel post-processing. Corroborating that observation, the AUROC results for the CoReSeg with U-Net as the backbone and Car as the UUC improved from 0.854 to 0.913 for the Vaihingen dataset, and from 0.768 to 0.816 for Potsdam. An improvement can be observed by analyzing the average AUROC for the 5 UUCs, which improved from

TABLE 7. The table shows Kappa scores with threshold values varying from 0.0 to 1.0 for the Vaihingen dataset with OpenGMM and OpenPCS as the OSS method and WRN50 as the backbone. The third column indicates the usage of the FuSC post-processing. The last two rows of each OSS method block show the average κ across all UUCs. For the rows with the average scores, the † symbol indicates if the results have statistical significance, according to a paired two-tailed t-Student test with $p \leq 0.05$ across the five (5) classes.

Method	UUC	FuSC	Thresholds									
			0.00	**0.10**	**0.20**	**0.30**	**0.40**	**0.50**	**0.60**	**0.70**	**0.80**	**0.90**
OpenGMM	Imp. Surf.	No	**0.500**	0.500	0.510	0.568	0.616	0.644	0.665	0.670	0.642	0.541
OpenGMM	Imp. Surf.	Yes	**0.500**	**0.530**	**0.560**	**0.591**	**0.621**	**0.649**	**0.676**	**0.693**	**0.687**	**0.617**
OpenGMM	Buildings	No	**0.461**	0.461	0.481	0.532	0.581	0.608	0.631	**0.649**	0.654	0.612
OpenGMM	Buildings	Yes	**0.461**	**0.493**	**0.524**	**0.555**	**0.586**	**0.615**	**0.643**	0.664	**0.678**	**0.652**
OpenGMM	Low Veg.	No	**0.606**	**0.575**	**0.567**	**0.551**	**0.525**	0.490	0.436	0.365	0.278	0.177
OpenGMM	Low Veg.	Yes	**0.606**	0.564	0.558	0.541	**0.525**	**0.496**	**0.462**	**0.405**	**0.325**	**0.218**
OpenGMM	High Veg.	No	**0.518**	0.509	0.523	0.534	0.536	0.520	0.484	0.418	0.304	0.166
OpenGMM	High Veg.	Yes	**0.518**	**0.512**	**0.526**	**0.537**	**0.539**	**0.525**	**0.498**	**0.443**	**0.368**	**0.217**
OpenGMM	Car	No	**0.775**	**0.747**	**0.712**	**0.672**	**0.607**	**0.515**	0.393	0.281	0.175	0.073
OpenGMM	Car	Yes	**0.775**	0.735	0.705	0.655	0.573	0.500	**0.414**	**0.324**	**0.223**	**0.131**
OpenGMM	Average	No	**0.572**	0.558	0.559	0.571	**0.573**	0.555	0.522	0.477	0.411	0.314
OpenGMM	Average	Yes	**0.572**	**0.567**	**0.575**	**0.576**	0.569	**0.557**	**0.539†**	**0.506†**	**0.456†**	**0.367†**
OpenPCS	Imp. Surf.	No	**0.500**	**0.525**	**0.552**	0.578	0.602	0.622	0.635	0.634	0.607	0.519
OpenPCS	Imp. Surf.	Yes	**0.500**	0.522	0.550	**0.583**	**0.613**	**0.638**	**0.656**	**0.661**	**0.648**	**0.575**
OpenPCS	Buildings	No	**0.461**	0.466	0.520	0.543	0.566	0.586	0.599	0.605	0.598	0.551
OpenPCS	Buildings	Yes	**0.461**	**0.472**	**0.524**	**0.551**	**0.579**	**0.607**	**0.627**	**0.632**	**0.640**	**0.603**
OpenPCS	Low Veg.	No	**0.606**	**0.606**	**0.604**	**0.589**	0.547	0.478	0.406	0.332	0.293	0.245
OpenPCS	Low Veg.	Yes	**0.606**	0.590	0.592	0.585	**0.562**	**0.532**	**0.463**	**0.367**	**0.310**	**0.254**
OpenPCS	High Veg.	No	**0.518**	0.511	0.529	0.541	0.542	0.527	0.489	0.424	0.319	0.168
OpenPCS	High Veg.	Yes	**0.518**	**0.517**	**0.541**	**0.554**	**0.556**	**0.540**	**0.505**	**0.456**	**0.379**	**0.213**
OpenPCS	Car	No	**0.775**	**0.771**	**0.763**	**0.732**	**0.623**	**0.468**	**0.369**	**0.293**	**0.198**	**0.109**
OpenPCS	Car	Yes	**0.775**	0.748	0.723	0.674	0.573	0.443	0.332	0.237	0.154	0.086
OpenPCS	Average	No	**0.572**	**0.576**	**0.594**	**0.597**	**0.576**	0.536	0.500	0.458	0.403	0.319
OpenPCS	Average	Yes	**0.572**	0.570	0.586	0.590	**0.576**	**0.552†**	**0.517†**	**0.471†**	**0.426†**	**0.346†**

0.850 to 0.880 (3.53%) for Vaihingen and from 0.764 to 0.797 (4.32%) for Potsdam.

Tables 7 and 4 show the comparison of Cohen's Kappa (κ) scores between different thresholds in the OSS methods tested. The κ score improved for all tested scenarios presented in Table 4. Figures followed by † show statistically significant improvements when using superpixel post-processing. For CoReSeg, we observed statistically significant improvements in 8 of the 11 scenarios, while for the OpenPCS and OpenGMM results shown in Table 7, the improvements were statistically significant for threshold values above 0.5. The proposed post-processing was able to improve the threshold independent AUROC score and the qualitative results. Furthermore, the results for Cohen's kappa score reassure that the post-processing improves the reliability of the predictions.

VI. CONCLUSION

This paper presented two approaches to improve known open-set semantic segmentation benchmarks for the Vaihingen and Potsdam datasets. The first one is OpenGMM, an extension of OpenPCS [13] that replaces the unimodal Principal Component model with a multimodal Gaussian Mixture of Models. The second is a superpixel post-processing pipeline capable of benefiting OSS predictions based on single SPS algorithms, or a mixture of them in a novel SPS fusion method named FuSC.

OpenGMM improved OpenPCS' AUROC average results for most of the UUCs and most of the tested backbones. Furthermore, the superpixel post-processing method achieved state-of-the-art results for both Vaihingen and Potsdam datasets when applied to CoReSeg's predicted segmentation. The improvement produced by the post-processing was statistically significant in most of the cases tested.

Our novel FuSC fusion scheme uses Malahanobis distance to merge neighboring segments below the established minimum size limit. FuSC produced more stable and reliable superpixel segmentation than the tested superpixel generation algorithms individually. Furthermore, in 11 of 12 cases, the best results were achieved using the final segmentation generated by FuSC. The FuSC method also relieved the burden of parameter selection for superpixel generation.

Our proposed superpixel post-processing method improved the results in all tested scenarios and for all OSS methods and backbones. This study shows that methods aimed at improving semantic consistency can benefit from a superpixel post-processing procedure, which helps in object and boundary delimitation.

We believe that due to the improvements achieved using post-processing in the segmentation pipeline, future works should include evaluating other contour recognition post-processing techniques. Besides, another approach to improve the segmentation may use a deep neural network that generates superpixels simultaneously with the semantic segmentation task [55], possibly using conditional random fields as the post-processing step at the end of the networks. Since the tested open-set methods use closed-set backbones, adapting different modern state-of-the-art

TABLE 8. Top 20 average AUROC results for all five UUCs tested scenarios achieved with different superpixel post-processing methods for OSS, obtained with CoReSeg [22] on the Vaihingen dataset. The first line shows the baseline result without post-processing. The first column presents the superpixel segmentation config used, the second shows which value was chosen to represent the superpixel to calculate the distance between 2 superpixels, the third the minimum pixel count for each superpixel, and the last column the average AUROC for the 5 tested scenarios.

Config	Value Used	min. size	UUCs					avg. AUROC
			Imp. Surf.	Buildings	Low Veg.	High Veg.	Car	
-	-	-	.884	.934	.710	.867	.854	.8499 ± .084
fusc03	mean	50	**906**	.959	.739	**.886**	.910	**.8800 ± .083**
fusc03	median	50	**.906**	.959	.739	**.886**	.910	**.8800 ± .083**
fusc03	mean	25	**.906**	.959	.739	**.886**	.910	.8797 ± .083
fusc03	median	25	.905	.959	.739	**.886**	.910	.8797 ± .083
fusc02	median	50	**.906**	.959	**.740**	.885	.908	.8796 ± .083
fusc01	mean	50	.904	**.960**	.739	.882	**.913**	.8796 ± .084
fusc02	mean	50	**.906**	.959	**.740**	.885	.908	.8796 ± .083
fusc01	median	50	.904	**.960**	.739	.882	**.913**	.8796 ± .084
fusc01	median	25	.904	**.960**	.738	.882	.912	.8794 ± .084
fusc01	mean	25	.904	**.960**	.738	.882	.912	.8794 ± .084
fusc02	median	25	**.906**	.959	.739	.885	.907	.8794 ± .083
fusc02	mean	25	**.906**	.959	.739	.885	.907	.8793 ± .083
fusc05	median	50	.905	**.960**	.739	.881	.910	.8789 ± .083
fusc06	mean	50	.905	.959	.738	.883	.909	.8788 ± .083
fusc05	mean	50	.905	**.960**	.739	.881	.909	.8788 ± .083
fusc06	median	50	.905	.959	.738	.883	.908	.8787 ± .083
fusc04	mean	50	.904	**.960**	.739	.880	.911	.8786 ± .083
fusc04	median	50	.904	**.960**	.739	.880	.911	.8786 ± .083
fusc05	median	25	.904	.959	.739	.881	.909	.8785 ± .083
fusc05	mean	25	.904	.959	.739	.881	.909	.8785 ± .083
fusc06	mean	25	.905	.959	.738	.883	.907	.8783 ± .083
fusc04	median	25	.903	**.960**	.739	.880	.910	.8783 ± .083

segmentation backbones [56] may improve the results. We can also conjecture that employing attention techniques would further improve the open-set segmentation.

APPENDIX A
SUPERPIXEL CONFIGURATIONS

Table 8 is complimentary to Section IV and shows the top 20 results for Vaihingen dataset and CoReSeg [22] among the 280 tests executed. The first line shows the baseline without post-processing, and the subsequent lines are sorted by Average AUROC. The second column stands for the value used to represent the superpixel for post-processing, and the third column shows the minimum pixel count of each superpixel used by FuSC.

Below the list of all superpixels configurations used to run all tests in this work presented in Sections V and IV and in Table 8. FZ stands for the Felzenszwalb and Huttenlocher [17] algorithm, QS stands for the Quickshift [36] algorithm, and SLIC stands for the method with the same name proposed by Achanta et al. [39]:

1) *single01*: FZ (scale: 100, sigma: 0.5, min_size: 50)
2) *single02*: FZ (scale: 200, sigma: 0.5, min_size: 50)
3) *single03*: SLIC (n_segments: n_pixels ÷ 350, compactness: 5, σ: 1)
4) *single04*: FZ (scale: 50, sigma: 0.5, min_size: 50)
5) *single05*: FZ (scale: 100, sigma: 0.5, min_size: 100)
6) *fusc01*:
   - SLIC (n_segments: n_pixels ÷ 2000, compactness: 5, σ: 1)
   - FZ (scale: 200, σ: 0.7, min_size: 200)
7) *fusc02*:
   - SLIC (n_segments: n_pixels ÷ 1500, compactness: 5, σ: 1)
   - FZ (scale: 100, σ: 0.7, min_size: 150)
8) *fusc03*:
   - SLIC (n_segments: n_pixels ÷ 1000, compactness: 5, σ: 1)
   - FZ (scale: 100, σ: 0.7, min_size: 150)
9) *fusc04*:
   - FZ (scale: 200, σ: 0.7, min_size: 200)
   - QS (kernel_size: 5, max_dist: 50, ratio: 0.5)
10) *fusc05*:
   - FZ (scale: 200, σ: 0.7, min_size: 200)
   - QS (kernel_size: 4, max_dist: 50, ratio: 0.5)
11) *fusc06*:
   - FZ (scale: 200, σ: 0.7, min_size: 200)
   - QS (kernel_size: 3, max_dist: 50, ratio: 0.5)

APPENDIX B
FUSING SUPERPIXELS FOR SEMANTIC CONSISTENCY - CODE

Listing 1 provides the complete code used for FuSC, implemented in Python 3.8. The comments detail the functioning of the method and the algorithmic complexity using big O notation.

The official implementation of all proposed approaches is available at https://github.com/iannunes/FuSC.

```
import scipy as sp
from scipy.spatial import distance

# method used to merge two different superpixel segmentations.
# s1 and s2: a 2D array mapping the superpixel segmentations. Each pixel in the represented in the segmentation array is set with the number of the respective segment
# img: the segmented image
# min_size: the minimum size threshold for the merged segmentation
def join_segmentations(s1, s2, img, min_size): #O(n)
  assert s1.shape == s2.shape
  counter = -1
  ret = np.zeros(s1.shape, dtype=int)
  final_labels = {}

  # merges two different segmentations. setting sequential labels to each intersection between s1 and s2
  for i in range(0, s1.shape[0]):  # O(n) - assuming constant time for dict operations
    for j in range(0, s1.shape[1]):
      label1 = s1[i,j]
      label2 = s2[i,j]
      if label1 not in final_labels:
        final_labels[label1]={}
      if label2 not in final_labels[label1]:
        final_labels[label1][label2] = counter
        counter-=1
      ret[i,j]=final_labels[label1][label2]

  ret = -1 * ret
  counter = -1 * counter
  existing_areas={}

  # ensure connectivity for each segment. O(n)
  for i in range(0, ret.shape[0]):
    for j in range(0, ret.shape[1]):
      label = ret[i,j]
      if label>0:
        if label in existing_areas:
          ret[ret==label] = counter
          # all operations are O(n) in the worst case when all labels must be replaced. This case is actually unfeasible.
          label = counter
          counter + 1
        existing_areas[label]=True
        ret = track_continuos(ret, i, j, label)
        # all operations are O(9n) in the worst case when all labels must be replaced. This case is actually unfeasible.

  ret = -1 * ret
  neighbors = {}
  # get the neighborhood for each segment
  for i in range(0, ret.shape[0]):     # O(n)
    for j in range(0, ret.shape[1]):
      label1 = ret[i,j]
      if label1 not in neighbors:
        neighbors[label1]={}

      for k in range(i-1,i+2):     # cte
        for h in range(j-1,j+2):
          if (k==h and k==0) or (k<0 or h<0 or k>=ret.shape[0] or h>=ret.shape[1]):
            continue

          label2 = ret[k,h]
          if label1 != label2:
            if label2 not in neighbors:
              neighbors[label2]={}
            neighbors[label1][label2]=True
            neighbors[label2][label1]=True

  return merge_superpixels(ret, neighbors, img, min_size)

# main procedure that merges neighboring areas if the minimum pixel count is not respected.
# sps: the 2D mapping superpixel segmentation
# neighbors: a list of each segment and its neighbors
# img: the segmented image
# min_size: the minimum size threshold for the merged segmentation
def merge_superpixels(sps, neighbors, img, min_size):
# the complexity for the procedure is 3O(n)+ O(n * cte * minimum size^2) + 2O(n) = O(n * minimum size^2). As some assumed constants depend on the minimum size, we can say that the procedure is pseudo-polynomial.
  sps_sizes={}

  img = np.array(img, dtype = float)
  sps_uniques = np.unique(sps, return_counts = True) # O(n)
  sps_processed = {}
  flatten_superpixels = {}

  # pixel count
  for i in range(0,len(sps_uniques[0])):   # O(n)
    sps_sizes[sps_uniques[0][i]] = sps_uniques[1][i]

  # populate a dictionary with image pixels for each segment
  for i in range(0, sps.shape[0]):     # O(n)
    for j in range(0, sps.shape[1]):
      label = sps[i,j]
      if label not in flatten_superpixels:
        flatten_superpixels[label] = []
      flatten_superpixels[label].append(img[i,j])

  for key in flatten_superpixels:
    flatten_superpixels[key] = np.array(flatten_superpixels[key])

  sps_mapping = OrderedDict()

  # for each segment with less pixels of minimum pixel count
  # compare to all neighbors and merge with closest one
  for key in flatten_superpixels:
    # O(n) as superpixel segmentation is an over segmentation of the image, the expected number of segments is n/cte implying in O(n/cte) = O(n) executions of the for loop
    if key in sps_mapping:
      continue
    while sps_sizes[key] < min_size:

      min_dist = 9999999999999999
      final_smaller_key = -1
      final_bigger_key = -1

      # closest neighbor search
      for n_key in neighbors[key]:
      # worst case scenario is n/2 iterations - O(n)
        # assuming that superpixel segmentations produces segments approximately with the same pixel count. And assuming the maximum number of possible neighbors for each segment, the number of iterations are 2 x minimum size + 6. We can consider as O(cte*min_size) = O (cte)
```

LISTING 1. FuSC implementation.

LISTING 1. *(Continued.)* FuSC implementation.

```
        if n_key in sps_mapping:
          continue
        smaller_sps_label = n_key
        bigger_sps_label = key
        if flatten_superpixels[key].shape[0] < flatten_superpixels[n_key].shape[0]:
          smaller_sps_label = key
          bigger_sps_label = n_key

        if final_smaller_key < 0:
          final_smaller_key = smaller_sps_label
          final_bigger_key = bigger_sps_label

        x = flatten_superpixels[smaller_sps_label]
        data = flatten_superpixels[bigger_sps_label]
        # computes the distance between the 2 segments
        dist = mahalanobis(x = np.mean(x,axis = 0), data = data)
        # the complexity of mahalanobis is the greatest between O((d**4)*((log d)**2)) and O(n*(d**2)), for n the number of elements in the biggest segment and d the number of features.
        # in our particular case, we have few features and the probable number of elements in the segment is cte*minimum size. The final complexity for our case is the greatest between O((cte**4)*((log cte)**2)) and O((cte*minimum size)*(cte**2)) = O(minimum size)

        if min_dist > dist:
          min_dist = dist
          final_smaller_key = smaller_sps_label
          final_bigger_key = bigger_sps_label

      # create the merging mapping. In the end, the mapping is executed to produces the final segmentation. O(cte)
      sps_mapping[final_smaller_key] = final_bigger_key
      # compute the size of the merged segment. O(cte)
      sps_sizes[final_bigger_key] = sps_sizes[final_smaller_key] + sps_sizes[final_bigger_key]

      for n_key in neighbors[final_smaller_key]:
        # as discussed before, the probable case is O(2*min_size) = O(cte)
        if final_smaller_key in neighbors[n_key]:
          del neighbors[n_key][final_smaller_key]
        neighbors[final_bigger_key][n_key]=True

      if final_smaller_key in neighbors[final_bigger_key]:
        del neighbors[final_bigger_key][final_smaller_key]
      if final_bigger_key in neighbors[final_bigger_key]:
        del neighbors[final_bigger_key][final_bigger_key]

      key = final_bigger_key

  sps = merge_mapped(sps, sps_mapping) #O(number of pixels)
  sps = relabel(sps)           #O(number of pixels)

  return sps

# auxiliary function to execute the relabel according to the intersection between the segmentations
# sps: the 2D mapping superpixel segmentation
# sps_mapping: a list maping merged superpixels
def merge_mapped(sps, sps_mapping):  # O(number of pixels)
  for i in range(0, sps.shape[0]):
    for j in range(0, sps.shape[1]):
      label = sps[i,j]
      while label in sps_mapping:
        sps[i,j] = sps_mapping[label]
        label=sps[i,j]
  return sps


# ensure that numbered lables are between 1 and n
# sps: the 2D mapping superpixel segmentation
def relabel(sps):  # O(number of pixels)
  counter = -1
  for i in range(0, sps.shape[0]):
    for j in range(0, sps.shape[1]):
      label = sps[i,j]
      if label<0:
        continue
      sps[sps==label] = counter
      counter -= 1
  return sps*-1

# recursive procedure that selects a continuous area and relabel it
def track_continuos(input_array, i, j, label, rec=False):
  if input_array[i, j] != label:
    return input_array

  input_array[i, j] = input_array[i, j]*-1
  for k in range(i-1,i+2):
    for h in range(j-1,j+2):
      if (k==h and k==0) or (k<0 or h<0 or k>=input_array.shape[0] or h>=input_array.shape[1]):
        continue
      input_array = track_continuos(input_array, k, h, label, rec=True)
  return input_array

# compute the mahalanobis distance between the 2 distributions.
# x: image pixels of a superpixel
# data: image pixels of a superpixel
# cov: pre calculated covariance matrix
def mahalanobis(x=None, data=None, cov=None):
  # O((n**4)*((log n)**2)) or O(N*(n**2))
  """Compute the Mahalanobis Distance between each row of x and the data
  x   : vector or matrix of data with, say, p columns.
  data : ndarray of the distribution from which Mahalanobis distance of each observation of x is to be computed.
  cov  : covariance matrix (p x p) of the distribution. If None, will be computed from data.
  """
  # N = number of data points in data
  # n = number of features in data

  x_minus_mu = x - np.mean(data,axis=0)
  if not cov:
    cov = np.cov(data.T)     # O(N*(n**2))
  inv_covmat = sp.linalg.inv(cov) # O((n**4)*((log n)**2))
  left_term = np.dot(x_minus_mu, inv_covmat) # O(n)
  m = np.dot(left_term, x_minus_mu.T)
  if type (m) is np.float64:
    return m
```

LISTING 1. *(Continued.)* FuSC implementation.

REFERENCES

[1] M. E. Paoletti, J. M. Haut, R. Fernandez-Beltran, J. Plaza, A. Plaza, J. Li, and F. Pla, ''Capsule networks for hyperspectral image classification,'' *IEEE Trans. Geosci. Remote Sens.*, vol. 57, no. 4, pp. 2145–2160, Apr. 2019.

[2] W. D. Ellis, *A Source Book of Gestalt Psychology*. Abingdon, Oxon, England: Routledge, 2013.

[3] D. Konstantinidis, ''Building detection for monitoring of urban changes,'' Ph.D. dissertation, Imperial College London, 2017.

[4] C. Almeida, W. Fernandes, S. Barbosa, and H. Lopes, ''Dealing with heterogeneous Google Earth images on building area detection task,'' in *Proc. Iberoamerican Congr. Pattern Recognit.*, Cham, Switzerland: Springer, 2018, pp. 133–140.

[5] X. Li, X. Yao, and Y. Fang, ''Building-A-Nets: Robust building extraction from high-resolution remote sensing images with adversarial networks,'' *IEEE J. Sel. Topics Appl. Earth Observ. Remote Sens.*, vol. 11, no. 10, pp. 3680–3687, Oct. 2018.

[6] E. Shelhamer, J. Long, and T. Darrell, ''Fully convolutional networks for semantic segmentation,'' *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 39, no. 4, pp. 640–651, Apr. 2017.

[7] O. Ronneberger, P. Fischer, and T. Brox, ''U-Net: Convolutional networks for biomedical image segmentation,'' in *Proc. 18th Int. Conf. Med. Image Comput. Comput.-Assist. Intervent. (MICCAI)*, Cham, Switzerland: Springer, 2015, pp. 234–241.

[8] V. Badrinarayanan, A. Kendall, and R. Cipolla, ''SegNet: A deep convolutional encoder–decoder architecture for image segmentation,'' *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 39, no. 12, pp. 2481–2495, Dec. 2017.

[9] S. Minaee, Y. Y. Boykov, F. Porikli, A. J. Plaza, N. Kehtarnavaz, and D. Terzopoulos, ''Image segmentation using deep learning: A survey,'' *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 44 no. 7, pp. 3523–3542, Feb. 2021.

[10] I. Ulku and E. Akagündüz, ''A survey on deep learning-based architectures for semantic segmentation on 2D images,'' *Appl. Artif. Intell.*, vol. 36, no. 1, pp. 1–45, Dec. 2022.

[11] R. Sekkal, C. Strauss, F. Pasteau, M. Babel, and O. Déforges, ''Fast pseudo-semantic segmentation for joint region-based hierarchical and multiresolution representation,'' in *Proc. 3rd Vis. Inf. Process. Commun.*, vol. 8305, Feb. 2012, pp. 156–162.

[12] C. Geng, S.-J. Huang, and S. Chen, ''Recent advances in open set recognition: A survey,'' *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 43, no. 10, pp. 3614–3631, Oct. 2021.

[13] H. Oliveira, C. Silva, G. L. S. Machado, K. Nogueira, and J. A. dos Santos, ''Fully convolutional open set segmentation,'' *Mach. Learn.*, vol. 112, no. 5, pp. 1733–1784, May 2023.

[14] M. Vendramini, H. Oliveira, A. Machado, and J. A. D. Santos, ''Opening deep neural networks with generative models,'' in *Proc. IEEE Int. Conf. Image Process. (ICIP)*, Sep. 2021, pp. 1314–1318.

[15] C. E. Rasmussen, *Gaussian Processes for Machine Learning*, in *Summer School on Machine Learning*. Cham, Switzerland: Springer, 2003, pp. 63–71.

[16] C. M. Bishop and N. M. Nasrabadi, *Pattern Recognition and Machine Learning*, vol. 4, no. 4. Cham, Switzerland: Springer, 2006.

[17] P. F. Felzenszwalb and D. P. Huttenlocher, ''Efficient graph-based image segmentation,'' *Int. J. Comput. Vis.*, vol. 59, no. 2, pp. 167–181, Sep. 2004.

[18] Q. Lin, W. Zhong, and J. Lu, ''Deep superpixel cut for unsupervised image segmentation,'' in *Proc. 25th Int. Conf. Pattern Recognit. (ICPR)*, Jan. 2021, pp. 8870–8876.

[19] T. Verelst, M. Blaschko, and M. Berman, ''Generating superpixels using deep image representations,'' 2019, *arXiv:1903.04586*.

[20] M. Wang, X. Liu, N. Q. Soomro, G. Han, and W. Liu, ''Content-sensitive superpixel segmentation via self-organization-map neural network,'' *J. Vis. Commun. Image Represent.*, vol. 63, Aug. 2019, Art. no. 102572.

[21] P. C. Mahalanobis, *On the Generalized Distance in Statistics*. Zafar Marg, DL, India: Nat. Inst.of Sci., 1936.

[22] I. Nunes, M. B. Pereira, H. Oliveira, J. A. dos Santos, and M. Poggi, ''Conditional reconstruction for open-set semantic segmentation,'' in *Proc. IEEE Int. Conf. Image Process. (ICIP)*, Oct. 2022, pp. 946–950.

[23] J. Ji, X. Lu, M. Luo, M. Yin, Q. Miao, and X. Liu, ''Parallel fully convolutional network for semantic segmentation,'' *IEEE Access*, vol. 9, pp. 673–682, 2021.

[24] L. Melas-Kyriazi and A. K. Manrai, ''PixMatch: Unsupervised domain adaptation via pixelwise consistency training,'' in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2021, pp. 12430–12440.

[25] J. Kang, Z. Wang, R. Zhu, X. Sun, R. Fernandez-Beltran, and A. Plaza, ''PiCoCo: Pixelwise contrast and consistency learning for semisupervised building footprint segmentation,'' *IEEE J. Sel. Topics Appl. Earth Observ. Remote Sens.*, vol. 14, pp. 10548–10559, Jan. 2021.

[26] R. Ratajczak, C. Crispim, B. Fervers, E. Faure, and L. Tougne, ''Semantic segmentation post-processing with colorized pairwise potentials and deep edges,'' in *Proc. 10th Int. Conf. Image Process. Theory, Tools Appl. (IPTA)*, Nov. 2020, pp. 1–6.

[27] Z. Hu, Q. Zou, and Q. Li, ''Watershed superpixel,'' in *Proc. IEEE Int. Conf. Image Process. (ICIP)*, Sep. 2015, pp. 349–353.

[28] H. Zhang, K. Jiang, Y. Zhang, Q. Li, C. Xia, and X. Chen, ''Discriminative feature learning for video semantic segmentation,'' in *Proc. Int. Conf. Virtual Reality Visualizat.*, Aug. 2014, pp. 321–326.

[29] M. Gunther, S. Cruz, E. M. Rudd, and T. E. Boult, ''Toward open-set face recognition,'' in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. Workshops (CVPRW)*, Jul. 2017, pp. 1757–1772.

[30] A. Bendale and T. E. Boult, ''Towards open set deep networks,'' in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 1563–1572.

[31] C. C. V. da Silva, K. Nogueira, H. N. Oliveira, and J. A. D. Santos, ''Towards open-set semantic segmentation of aerial images,'' in *Proc. IEEE Latin Amer. GRSS ISPRS Remote Sens. Conf. (LAGIRS)*, Mar. 2020, pp. 16–21.

[32] S. Prasad, B. Le Saux, N. Yokoya, and R. Hansch. (2020). *IEEE GRSS Data Fusion Challenge*. [Online]. Available: https://dx.doi.org/10.21227/jnh9-nz89

[33] Z. Cui, W. Longshi, and R. Wang, ''Open set semantic segmentation with statistical test and adaptive threshold,'' in *Proc. IEEE Int. Conf. Multimedia Expo. (ICME)*, Jul. 2020, pp. 1–6.

[34] J. Cen, P. Yun, J. Cai, M. Yu Wang, and M. Liu, ''Deep metric learning for open world semantic segmentation,'' in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2021, pp. 15313–15322.

[35] J. Hong, W. Li, J. Han, J. Zheng, P. Fang, M. Harandi, and L. Petersson, ''GOSS: Towards generalized open-set semantic segmentation,'' 2022, *arXiv:2203.12116*.

[36] A. Vedaldi and S. Soatto, ''Quick shift and kernel methods for mode seeking,'' in *Proc. Comput. Vis. (ECCV) 10th Eur. Conf. Comput. Vis.*, Cham, Switzerland: Springer, 2008, pp. 705–718.

[37] A. Levinshtein, A. Stere, K. N. Kutulakos, D. J. Fleet, S. J. Dickinson, and K. Siddiqi, ''TurboPixels: Fast superpixels using geometric flows,'' *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 31, no. 12, pp. 2290–2297, Dec. 2009.

[38] M.-Y. Liu, O. Tuzel, S. Ramalingam, and R. Chellappa, ''Entropy rate superpixel segmentation,'' in *Proc. CVPR*, Jun. 2011, pp. 2097–2104.

[39] R. Achanta, A. Shaji, K. Smith, A. Lucchi, P. Fua, and S. Süsstrunk, ''SLIC superpixels compared to state-of-the-art superpixel methods,'' *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 34, no. 11, pp. 2274–2282, Nov. 2011.

[40] P. Morerio, L. Marcenaro, and C. S. Regazzoni, ''A generative superpixel method,'' in *Proc. 17th Int. Conf. Inf. Fusion*, Jul. 2014, pp. 1–7.

[41] P. Buyssens, M. Toutain, A. Elmoataz, and O. Lézoray, ''Eikonal-based vertices growing and iterative seeding for efficient graph-based segmentation,'' in *Proc. IEEE Int. Conf. Image Process. (ICIP)*, Oct. 2014, pp. 4368–4372.

[42] M. V. D. Bergh, X. Boix, G. Roig, B. D. Capitani, and L. V. Gool, ''SEEDS: Superpixels extracted via energy-driven sampling,'' in *Proc. Comput. Vis. (ECCV) 12th Eur. Conf. Comput. Vis.*, Cham, Switzerland: Springer, 2012, pp. 13–26.

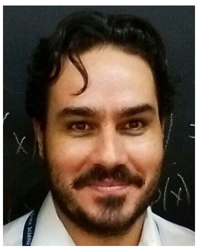
[43] Z. Li and J. Chen, ''Superpixel segmentation using linear spectral clustering,'' in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2015, pp. 1356–1363.

[44] V. Machairas, M. Faessel, D. Cárdenas-Peña, T. Chabardes, T. Walter, and E. Decenciére, ''Waterpixels,'' *IEEE Trans. Image Process.*, vol. 24, no. 11, pp. 3707–3716, Nov. 2015.

[45] A. Rubio, L. Yu, E. Simo-Serra, and F. Moreno-Noguer, ''BASS: Boundary-aware superpixel segmentation,'' in *Proc. 23rd Int. Conf. Pattern Recognit. (ICPR)*, Dec. 2016, pp. 2824–2829.

[46] R. Achanta, P. Márquez-Neila, P. Fua, and S. Süsstrunk, ''Scale-adaptive superpixels,'' in *Proc. 26th Color Imag. Conf. Final Program*, no. 1, 2018, pp. 1–6.

[47] J. Zhang, P. Wang, F. Gong, H. Zhu, and N. Chen, "Content-based superpixel segmentation and matching using its region feature descriptors," *IEICE Trans. Inf. Syst.*, vol. 103, no. 8, pp. 1888–1900, 2020.
[48] E. Elkhateeb, H. Soliman, A. Atwan, M. Elmogy, K.-S. Kwak, and N. Mekky, "A novel coarse-to-fine sea-land segmentation technique based on superpixel fuzzy C-Means clustering and modified chan-vese model," *IEEE Access*, vol. 9, pp. 53902–53919, 2021.
[49] S. Lloyd, "Least squares quantization in PCM," *IEEE Trans. Inf. Theory*, vol. IT-28, no. 2, pp. 129–137, Mar. 1982.
[50] X. Sun, Z. Yang, C. Zhang, K.-V. Ling, and G. Peng, "Conditional Gaussian distribution learning for open set recognition," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2020, pp. 13477–13486.
[51] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 2261–2269.
[52] S. Zagoruyko and N. Komodakis, "Wide residual networks," in *Proc. Brit. Mach. Vis. Conf.*, 2016, pp. 924–928.
[53] A. Paszke et al., "Pytorch: An imperative style, high-performance deep learning library," in *Proc. Adv. Neural Inf. Process. Syst.*, 2019, pp. 8024–8035. [Online]. Available: http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf
[54] J. Cohen, "A coefficient of agreement for nominal scales," *Educ. Psychol. Meas.*, vol. 20, no. 1, pp. 37–46, Apr. 1960.
[55] F. Yang, Q. Sun, H. Jin, and Z. Zhou, "Superpixel segmentation with fully convolutional networks," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2020, pp. 13961–13970.
[56] J. Wang, K. Sun, T. Cheng, B. Jiang, C. Deng, Y. Zhao, D. Liu, Y. Mu, M. Tan, X. Wang, W. Liu, and B. Xiao, "Deep high-resolution representation learning for visual recognition," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 43, no. 10, pp. 3349–3364, Apr. 2019.



IAN MONTEIRO NUNES received the master's and Ph.D. degrees in informatics from the Department of Informatics, Pontifical Catholic University of Rio de Janeiro (PUC-Rio), Rio de Janeiro, Brazil. He is currently with the Brazilian Institute of Geography and Statistics (IBGE) as the Head of the research group on Earth observation applied to official agricultural statistics and the agricultural census; a Data Science Professor with the National School of Statistics (ENCE); a member of the UN-CEAG/UN-CEBD Joint Task Team on the use of Earth observation data for agricultural statistics; and a Researcher with the Big Data and Oficial Statistics Research Group (http://dgp.cnpq.br/dgp/espelhorh/5909149338114540), ENCE. He is also a Computer Engineer with the Department of Informatics, PUC-Rio. His research interests include image segmentation, machine learning, deep learning, domain adaptation, computer vision, remote sensing, clustering, and open set recognition and segmentation.

MATHEUS B. PEREIRA received the bachelor's degree in computer science from Universidade Federal de Lavras (UFLA) and the master's degree in computer science from Universidade Federal de Minas Gerais (UFMG). He is currently pursuing the joint Ph.D. degree in computer science with UFMG and Université Gustave Eiffel, France. His research interests include computer vision, pattern recognition, and machine/deep learning, mainly applied to remote sensing data analysis.

HUGO OLIVEIRA received the bachelor's and master's degrees in computer science from Universidade Federal da Paraíba (UFPB) and the Ph.D. degree in computer science from Universidade Federal de Minas Gerais (UFMG). Currently, he is a Professor with the Informatics Department (DPI), Universidade Federal de Viçosa (UFV), and a Postdoctoral Fellow with the Institute of Mathematics and Statistics (IME), Universidade de São Paulo (USP). His research interests include computer vision, pattern recognition, and machine/deep learning, mainly applied to medical imaging, remote sensing, and geological data analysis.

JEFERSSON ALEX DOS SANTOS received the bachelor's degree in computer science from the University of Mato Grosso do Sul (UEMS), Brazil, in 2006, the master's degree in computer science from the University of Campinas (Unicamp), Brazil, in 2009, and the joint Ph.D. degree in computer science from Unicamp and the University of Cergy-Pontoise, France, in 2013. He was an Associate Professor with Universidade Federal de Minas Gerais (UFMG), Brazil, from 2013 to 2022. He is currently an Assistant Professor (a Lecturer) of computing science with The University of Sheffield, England, U.K. He is also the Founder of the Laboratory of Pattern Recognition and Earth Observation (PATREO), Department of Computer Science, UFMG. His research interests include remote sensing image processing, computer vision, and machine learning. He serves as a PC member for several refereed journals and conferences and is currently the Vice President of the IEEE GRSS Brazilian Chapter. He used to hold the prestigious CNPq Productivity Research Scholarship, from 2016 to 2022. In 2021, he received Competitive Research Grant from the Serrapilheira Institute, Brazil. He is currently an Associate Editor of IEEE Geoscience and Remote Sensing Letters and the Co-Chair of the ISPRS Working Group for AI/ML for Geospatial Data.